НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

До захисту допущено:
Завідувач кафедри
_____ *Сергій СТІРЕНКО*
(підпис)
“___”_____20__ р

Дипломний проект

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Комп'ютерні системи та мережі»

спеціальності 123 «Комп’ютерна інженерія»

на тему: «Голосове управління комп’ютером на основі глосарію за допомогою алгоритмів розпізнавання мови»

Виконав:
студент _IV_курсу, групи__ІО-61__

___Дудченко Ігор Віталійович_____ _____
(прізвище, ім’я, по батькові) (підпис)

Керівник _доцент к.т.н. Верба Олександр Андрійоич___ _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Консультант __н. контроль_ _професор Сімоненко В.П._ _____
(назва розділу) (вчені ступінь та звання, прізвище, ініціали) (підпис)

Рецензент __доцент к.т.н. Орлова Марія Миколаївна____ _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2020 року

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМ. ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 123 «Комп'ютерна інженерія»

Освітньо-професійна програма «Комп'ютерні системи та мережі»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ *Сергій СТІРЕНКО*
(підпис)
"___"_____20__ р.

ЗАВДАННЯ

на дипломний проект студенту

Дудченку Ігорю Віталийовичу

1. Тема проекту «*Голосове управління комп'ютером на основі глосарію за допомогою алгоритмів розпізнавання мови»*

керівник проекту Верба Олександр Андрійович, доцент к.т.н., затверджені наказом по університету від *« 07 » травня* 2020р. № 1081-с

2. Термін здачі студентом закінченої роботи *2020р.*

3. Вихідні дані до проекту *технічне завдання, теоретичні дані.*

4. Зміст пояснювальної записки: *опис предметної області, опис проблеми, існуючих рішень, формулювання власного рішення, вибір та порівняння засобів для написання додатку, опис архітектури та графічного інтерфейсу додатку.*

5. Консультант роботи, з вказівкою розділів роботи, які до них вносяться

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Нормоконтроль	Сімоненко В.П.		

6. Дата видачі завдання *01.09 .2019 року*

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Найменування етапів дипломного проекту (роботи)	Строк виконання етапів проекту(роботи)	Примітки
1.	*Затвердження теми роботи*	01.09.2019	
2.	*Вивчення та аналіз завдання*	15.09.2019	
3.	*Розробка архітектури та загальної структури систем*	04.10.2019	
4.	*Розробка структур окремих підсистем*	13.12.2019	
5.	*Програмна реалізація системи*	03.02.2020	
6.	*Оформлення пояснювальної записки*	04.04.2020	
7.	*Передзахист*	26.05.2020	
8.	*Захист*	25.06.2020	

Студент Ігор ДУДЧЕНКО _____

(підпис)

Керівник Олександр ВЕРБА _____

(підпис)

Анотація

В бакалаврському дипломному проекті розроблено програмний додаток для голосово управління комп’ютером на основі глосарію за допомгою алгоритмів розпізнавання мови. Запропоноване рішення проблеми дозволяє додавати нові команди та відповідні дії для них у глосарій, завдяки чому голосового асистента можна змінювати під власні потреби. Продукт був створений для Linux-подібних систем за допомогою використання Google Cloud Speech API, на мові Python з графічним інтерфейсом написаним за допомогою Qt.

Annotation

The bachelor's thesis project developed a software application for voice control of a computer based on a glossary with the help of language recognition algorithms. The proposed solution allows you to add new commands and appropriate actions for them to the glossary, so you can change the voice assistant to suit your needs. The product was created for Linux-like systems using the Google Cloud Speech API, in Python with a graphical user interface written in Qt.

ВІДОМІСТЬ ДИПЛОМНОГО ПРОЕКТУ

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проект	2	
2	А4	ІАЛЦ.045430.001 ВП	Відомість проекту	1	
3	А4	ІАЛЦ.045430.002 ТЗ	Технічне завдання	4	
4	А4	ІАЛЦ.045430.003 ПЗ	Пояснювальна записка	60	
5	А3	ІАЛЦ.045430.004 Д1	Принципова схема додатку голосового помічника	1	
6	А3	ІАЛЦ.045430.005 Д2	Структурна схема додатку голосового помічника	1	
7	А3	ІАЛЦ.045430.006 Д3	Функціональна схема алгоритму роботи голосового помічника	1	
8	А4	ІАЛЦ.045430.007 Д4	Текст програми	18	

					ІАЛЦ.045430.001 ВП	
Зм.		*№ документа*	*Підпис*	*Дата*		
Розробив		*Дудченко І.В.*			Відомість дипломного проекту	*Літ.* / *Аркуш* / *Аркушів*
Перевірив		Верба О.А				1 / 1
						НТУУ «КПІ ім. Ігоря Сікорського» ФІОТ гр. ІО-61
Н. Контр.		*Сімоненко В.П.*				
Затвердив		*Стіренко* С.Г.				

ТЕХНІЧНЕ ЗАВДАННЯ

до дипломної роботи
освітньо-кваліфікаційного рівня___бакалавр___

на тему: “ Голосове управління комп’ютером на основі глосарію за допомогою алгоритмів розпізнавання мови ”

Київ – 2020 року

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ

Дане технічне завдання поширюється на розробку курсу «Інженерія програмного забезпечення». Область застосування: практичне використання людьми при користуванні комп’ютерними засобами.

2. ПІДСТАВИ ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на виконання роботи кваліфікаційно-освітнього рівня «бакалавр комп'ютерної інженерії», затверджене кафедрою обчислювальної техніки Національного технічного Університету України «Київський Політехнічний інститут ім. Ігоря Сікорського».

3. МЕТА І ПРИЗНАЧЕННЯ РОЗРОБКИ

Метою даного проекту є розробка голосового помічника для голосового управління комп’ютером на основі глосарію за допомогою алгоритмів розпізнавання мови.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелом розробки є науково-технічна література з теорії і практики програмування, бакалаврські роботи інших студентів, публікації в Інтернеті з даних питань.

5. **ТЕХНІЧНІ ВИМОГИ**

5.1. Вимоги до розробляємого продукту

- Асистент повинен виконувати будь-які дії, вказані в глосарії.
- Система повинна мати можливість додавати нові ключові слова та відповідні до них дії в глосарій.

5.2. Вимоги до програмного забезпечення

- Linux-подібна операційна система.
- Наявність на комп’ютері Python не нижче версії 3.0.

5.3. Вимоги до апаратної частини

- Оперативної пам’яті не менше 512 Мбайт.
- Вільне місце на жорсткому диску не менше 500 Мбайт.
- Звукова карта
- Мікрофон
- Процессор 1.4 ГГц

6. ЕТАПИ РОЗРОБКИ

	Дата
Вивчення літератури	28.03.2020
Складання і узгодження технічного завдання	03.04.2020
Створення модулів системи, що розробляється	15.04.2020
Тестування окремих модулів системи	25.04.2020
Допрацювання, налагодження і виправлення помилок	01.05.2020
Оформлення документації дипломної роботи	15.05.2020

ПОЯСНЮВАЛЬНА ЗАПИСКА
до дипломного проекту
на тему: «Голосове управління комп’ютером на основі глосарію за допомогою алгоритмів розпізнавання мови»

Київ – 2020 року

ЗМІСТ

ВСТУП

В даний час комп’ютерна техніка використовується в багатьох областях людської діяльності, будучи зручним і багатофункціональним інструментом. Однак, в даний час користувачі персонального комп'ютера (ПК) змушені використовувати способи взаємодії, слабо адаптовані до можливостей людського спілкування і обмежують здібності людини до обміну інформацією.

Персональний комп'ютер — електронна комп'ютерна машина, що призначена для зберігання і обробки інформації. Ціна ПК, його розміри та можливості задовольняють потреби багатьох людей.

Основна мета удосконалення та розвитку інтерфейсу людина-комп'ютер полягає в організації обміну інформацією з ПК таким чином, щоб:

- знизити час освоєння програмних і апаратних засобів;
- знизити рівень помилок при передачі інформації;
- зробити роботу з ПК можливою для людей, які не мають можливості користуватися традиційними засобами інтерфейсу;
- знизити стомлюваність, збільшити суб'єктивне задоволення користувача від роботи.

Для досягнення поставлених цілей необхідно застосування засобів взаємодії, більш повно використовують комунікативні здібності людини. Людина наділена великою кількістю можливостей сприймати і передавати інформацію: зір, слух (в тому числі усне мовлення), жести і рухи, міміка, дотик і іншими. У взаємодії людини і комп'ютера існують два інформаційних потоку:

- керуючі команди і дані, передані комп'ютера для обробки;
- результати обчислень та інша інформація , яка надається комп'ютером користувачеві.

Поширений в даний час людино-машинний інтерфейс відображає дані у вигляді умовних знаків на екрані комп'ютера. Користувач сприймає інформацію природними візуальним шляхом та виконує команди через натискання конкретних клавиш.

Інтерфейс засобом розпізнавання мови, жестів, міміки мало розповсюджені.

Особа людини є важливим джерелом інформації при спілкуванні між людьми. Вираз обличчя, міміка, артикуляція при розмові, рухи головою є зручним, природним і, що важливо, не обтяжливим способом передачі інформації. Нездатність комп'ютера з одного боку сприйняти, а з іншого боку відтворити настільки природні для людини способи спілкування ускладнює передачу і сприйняття інформації при роботі з ПК.

Для забезпечення ефективного мовного діалогу між користувачем і ПК необхідні стійкі системи розпізнавання мови.

Метою дипломного проекту «Голосове управління комп'ютером на основі глосарію за допомогою алгоритмів розпізнання мови» є створення програми для використання у повсякденному житті. Об’єктом автоматизації є розпізнавання мови на основі глосарію.

Часто пошук інформації є дуже не зручним та повільним для користувача ПК. Постає потреба у створенні комп’ютерної програми, яка буде, швидко виконувати пошук інформації засобом голосу, а також автоматизувати виконання рутинних команд, що виникають під час пошуку інформації в мережі Інтернет за допомогою клавіатури.

РОЗДІЛ 1
ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

1.1 Автоматичне розпізнавання мови

Щодо технологічного розвитку, ми, можливо, ще пару десятків років від наявності по-справжньому автономних, розумних систем що спілкуються з нами по-справжньому «по-людськи». Однак багато в чому ми прогресуємо неухильно завдяки постійному розвитку технологій автоматизованого розпізнавання мовлення.

Автоматичне розпізнавання мови (АРМ) - це технологія, яка дозволяє людям використовувати свої голоси, щоб взаємодіяти з комп'ютерним інтерфейсом таким чином, що у своїх найскладніших варіантах нагадує звичайну людську розмову.

В даний час найкращими прикладами розроблених технологій АРМ є Siri на від компанії Apple; Alexa від Amazon; Cortana від Windows та інші системи, що використовуються в бізнес та передових технологіях. Однак навіть ці програми, незважаючи на "точність" приблизно від 96 до 99%, можуть досягти таких результатів лише в ідеальних умовах та для конкретних ключових слів.

Основна послідовність подій, за допомогою яких будь-яке програмне забезпечення АРМ, незалежно від його витонченості, підбирає та розбиває ваші слова для аналізу та відповіді:

- Ви розмовляєте з програмним забезпеченням через аудіо-канал
- Пристрій, який ви використовуєте створю аудіо-хвилі сказаних вами звуків
- Хвилі очищаються шляхом видалення фонового шуму та нормалізації гучності
- Отримана відфільтрована хвильова форма потім розпадається на так звані фонеми
- Кожна фонема схожа на ланцюжок, і програмне забезпечення АСР аналізує їх послідовно, починаючи з першої фонеми, використовуючи статистичний аналіз ймовірності для виведення цілих слів, а звідти - повних речень
- Ваш система розпізнавання мови "зрозумівши" ваші слова, може виконати відповідні дії

1.2 Методи розпізнавання мови

Розпізнавання мови та її переклад у цифровий текстовий варіант - це сфера, якою займаються багато програмістів вже кілька років. Це широка область, яка містить у собі багато наукових дисциплін, такі як лінгвістика, математика та інформатика. Для досягнення правильних результатів потрібна складна діагностика голосу

Розпізнавання мови зазвичай застосовується до служб, де не зручно користуватися звичайними методами введення. Також це широко застосовують у військових, особливо у військово-повітряних силах, транспорті, освіті або автономних системах (найпоширеніші види використання програмного забезпечення розпізнавання голосу).

Існує багато способів створити додаток, використовуючи деякі API промови до тексту. Найбільші компанії, такі як Google, Microsoft, Amazon та Apple, використовують власні алгоритми для обробки голосу та викладають багато ресурсів для їх вдосконалення. Ці послуги не стосуються запису голосу, а лише намагаються перенести його у текстову форму.

Алгоритми розпізнавання мовлення дуже складні, це одна із причин, чому використовується штучний інтелект.

Існує три найбільш використовувані методи розпізнавання мовлення:

- Прихована марковська модель (ПММ)
- Алгоритм динамічної трансформації тимчасової шкали (АДТТШ)
- Нейронні мережі

Прихована марковська модель - це тип графічної моделі, яка часто використовується для моделювання часових даних. ПММ припускають, що спостережувані дані не є фактичним станом моделі, але натомість генерується основними прихованими станами. Через гнучкість і обчислювальну ефективність, ПММ знайшли широке застосування у багатьох сферах. Вони відомі тим, що використовують їх у розпізнаванні та генерації тимчасових зразків, таких як розпізнавання мовлення, розпізнавання рукописного тексту та синтез мовлення.

Алгоритм динамічної трансформації тимчасової шкали - це алгоритм вимірювання подібності між двома послідовностями, які можуть змінюватися за часом або швидкістю. Можна порівняти його з двома відео записами. Наприклад, подібність у моделях ходьби буде виявлена, навіть якщо в одному відео людина йшла повільно, а в іншому відео, людина йшла швидше, або навіть якщо під час одного були прискорення та уповільнення спостереження. Взагалі, АДТТШ - це метод, який дозволяє комп'ютеру знайти оптимальний збіг між двома заданими послідовностями з певними обмеженнями. Простіше кажучи, в той час як ПММ використовує механізм розбору голосового сигналу та ймовірності входження частин речення, АДТТШ використовує розпізнавання окремих слів.

Нейронні мережі здатні вирішувати складніші завдання розпізнавання, але не так добре, як ПММ, якщо мова йде про великі словникові запаси. Нейроні мережеві технології використовуються через наступні причини: це зменшує блок моделювання, щоб підвищити рівень розпізнавання, також є глибинне навчання, яке можна використовувати для розробки комбінації гібридної системи.

1.3 Application Programming Interfaces

Веб-сервіси - загальноприйняте рішення, щоб забезпечити єдиний інтерфейс програмування до багатьох мов. Клієнтські програми використовують інтерфейси програмування прикладних програм (API) для спілкування з веб-службами. Взагалі кажучи, API розкриває набір даних та функції для полегшення взаємодії між клієнтом та веб-сервером. Як показано на рис. 1.1, веб-API - це обличчя інтернет сервісу, безпосередньо прослуховування та відповідь на запити клієнта.

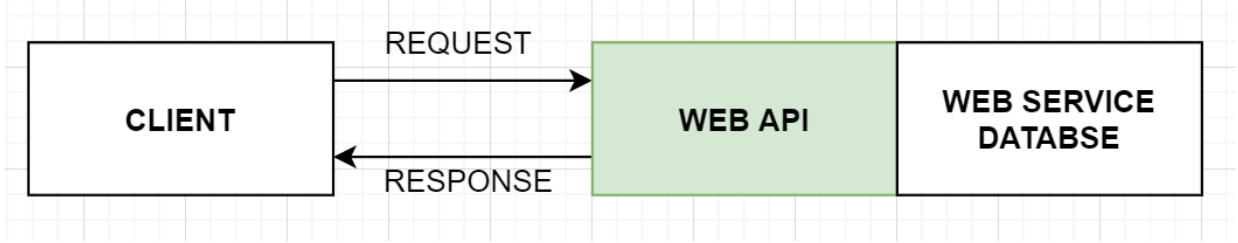


Рис. 1.1 Схема роботи API

З моменту свого впровадження в 2000 році [8] архітектурний стиль Representational State Transfer (REST) все частіше застосовується до дизайну API для сучасної мережі послуг. SOAP - також популярна технологія, яка використовується для цього [9], але обтяжена значними налаштуваннями та обробкою накладних витрат для клієнта. REST, навпаки, заохочує повторне використання технології HTTP для надсилання та отримання даних таким же чином, як веб-браузер запитує та отримує веб-сторінку за допомогою уніфікованих локаторів ресурсів (URL-адрес).

Нарешті, REST архітектура не накладає обмежень щодо формату на повернені дані. Маючи REST API робить веб-сервіс "RESTful"

Добре розроблені REST API можуть залучати розробників клієнтів до використання веб-служб. На сьогоднішньому відкритому ринку, на якому конкурентні веб-сервіси змагаються за увагу, естетично приємний дизайн REST API - обов’язкова функція. Для розширення доступу до REST було розроблено ряд власних сторонніх API таких мов, як Python, R та JavaScript.

1.4 Хмарні обчислення

Хмарні обчислення (Cloud computing) - це доставка різних сервісів через Інтернет. До таких ресурсів належать такі інструменти та програми, як зберігання даних, сервери, бази даних, мережа та програмне забезпечення.

Замість того, щоб зберігати файли на власному жорсткому диску або локальному пристрої зберігання даних, хмарне зберігання дозволяє зберігати їх у віддаленій базі даних. Поки електронний пристрій має доступ до Інтернету, він має доступ до даних та програмних програм для їх запуску.

Хмарні обчислення - це популярний варіант для людей та підприємств з ряду причин, включаючи економію витрат, підвищення продуктивності, швидкості та ефективності, продуктивності та безпеки

Хмарні обчислення названі такими, тому що інформація, до якої звертаються, знаходиться віддалено в хмарі або віртуальному просторі.

Компанії, що надають хмарні послуги, дозволяють користувачам зберігати файли та програми на віддалених серверах, а потім отримувати доступ до всіх даних через Інтернет. Це означає, що користувачеві не потрібно знаходитись у певному місці, щоб отримати доступ до нього, що дозволяє користувачеві працювати віддалено.

Вони можуть бути як державними, так і приватними. Громадські хмарні служби надають свої послуги через Інтернет за певну плату. Приватні хмарні сервіси, з іншого боку, надають послуги лише певній кількості людей. Ці сервіси - це система мереж, що постачають розміщені послуги. Існує також гібридна опція, яка поєднує в собі елементи як державних, так і приватних послуг.

1.5 API для розпізнавання мови

Сьогодні багато великих компаній надають API для виконання різних завдань машинного навчання. Розпізнавання мови не є винятком. Щоб використовувати ці API, вам не потрібно бути спеціалістом з обробки мов. Зазвичай вони забезпечують зручний інтерфейс. Все, що вам потрібно зробити - це надіслати HTTP-запит із необхідним вмістом на сервер API. Тоді ви отримаєте відповідь із виконаними завданнями. Цей підхід корисний, коли вам не потрібно щось особливе. Іншими словами, якщо ваша проблема є стандартною і відомою. Ще одна перевага цього способу полягає в тому, що ви можете заощадити такі цінні ресурси, як час і гроші.

Тим не менш, існує багато ситуацій, коли ви не можете використовувати API і вам потрібно розробити систему розпізнавання мови з нуля. Цей спосіб досить складний, вимагає багато зусиль та ресурсів, але в результаті ви можете створити систему, яка буде ідеально сумісна з вашими потребами. Також можна покращити якість результатів, якщо побудувати алгоритми самостійно. У всякому разі, добре знати про API. Ви можете зрозуміти, що може зробити кожен API, які плюси та мінуси вони мають тощо. Отже, ви зможете визначити, коли ви повинні використовувати API (і який API) і коли ви повинні думати про свою власну систему.

Є два основні завдання в обробці мовлення.

По-перше, це перетворення мови в текст. Друге - перетворити текст у людську мову. Ось перелік деяких популярних API для обробки мови:

- Google Cloud Speech API
- IBM Watson Speech to Text
- Microsoft Azure Bing Speech API
- Amazon Transcribe
- VoxSigma API
- Alexa Voice Service

В цьому розділі буде детальніше розглянутий кожен сервіс.

1.5.1 IBM Watson Speech to Text

IBM Watson Speech to Text - це послуга, що надається IBM Watson, яка може перетворити людську мову в текст. IBM Watson підтримує налаштування не тільки для конкретного словника слів, але і для конкретного акустичного стану. Отже, ви можете адаптувати систему до середовища, де ви плануєте її використовувати. Основна вада IBM Watson Speech to Text - це дуже мала кількість підтримуваних мов. Крім того, спеціальні моделі доступні для ще меншої кількості мов. Інші корисні функції, доступні в програмі IBM Watson Speech to Text, - це альтернативи слова (у бета-версії), часові позначки слова, фільтрація нецензурної лексики, розумне форматування номерів телефонів, дат, валюти тощо.

1.5.2 Microsoft Azure Bing Speech API

Microsoft Azure Bing Speech API - це компонент хмарних сервісів Microsoft Azure, що дозволяє одночасно вирішувати дві задачі: конвертування мову в текст, а також перетворення тексту в мовлення.

Мовлення перетворюється до тексту в API Speech Azure Bing в режимі реального часу, також можна налаштовувати, форматувати текст, фільтрувати нецензурну лексику, нормалізувати текст. Також є інтеграція з Azure LUIS. Azure LUIS дозволяє витягувати значення з тексту, а також основних об'єктів.

API є безкоштовним при здійснюванні до 5000 транзакцій на місяць.

Якщо вам потрібно більше, ви повинні платити 4 USD за кожні 1000 транзакцій.

1.5.3 Amazon Transcribe

Amazon Transcribe є частиною інфраструктури Amazon Web Services. Ви можете проаналізувати свої аудіодокументи, що зберігаються в Amazon S3, і отримати текст, зроблений з аудіо. Amazon Transcribe може додавати розділові знаки та форматування тексту. Ще одна цінна функція, яку надає ця послуга, - це підтримка звуку телефонії. Це тому, що аудіо з телефонних розмов часто має низьку якість. Система додає часові позначки для кожного слова в тексті. Отже, ви зможете зіставити кожне слово в тексті з відповідним місцем в аудіофайлі. Очікується, що API незабаром зможе розпізнати декілька спікерів та позначити їх голоси в тексті. Створення власних слів також э доступни. Користувачі зможуть явно додати, наприклад, назви своїх продуктів чи якісь інші конкретні слова. Існує безкоштовний рівень цін: ви можете користуватися послугою безкоштовно протягом перших 12 місяців після реєстрації (максимум 60 хвилин аудіо в місяць). Після цього періоду вам потрібно буде заплатити 0,0004 USD за секунду обробленого аудіо.

1.5.4 VoxSigma API

VoxSigma API - продукт компанії Vocapia Research. Ця компанія спеціалізується на галузі мовних технологій. VoxSigma API може не тільки конвертувати вхідну мову в текст, але й виконувати ідентифікацію мови та вирівнювання мови та тексту. Інші цікаві особливості API полягають у тому, що він може додавати розділові знаки до вихідного тексту, обчислювати оцінку достовірності виводу. Також VoxSigma API може обробляти числові та деякі інші об'єкти (наприклад, валюти) унікальним способом.

Можна настроїти власну мовну модель, але для цього вам потрібно зв’язатися з компанією та поговорити з ними безпосередньо.

Компанія пропонує кілька планів використання. Найпопулярніший метод - це похвилинна оплата.

Згідно з цим планом, вам доведеться платити 0,01 USD (або EUR) за хвилину. Цікаво, що вони враховують лише місця на вхідному аудіо, де присутня деяка мова. Іншими словами, якщо у вашому вхідному аудіо є кілька безшумних місць, тривалість цих місць буде відраховуватися під час обчислення загальної вартості. Безкоштовний пробний період також доступний, але для цього вам потрібно зв’язатися безпосередньо з компанією.

1.5.5 Alexa Voice Service

Alexa Voice Service (AVS) - це пристрої, створені за допомогою сервісу Alexa Voice Service (AVS), які мають мікрофон та динамік. Ви можете спілкуватися з цим пристроєм безпосередньо за допомогою слова "Alexa" та отримувати голосові відповіді та вміст миттєво. API Alexa Voice Service дає можливість створити “розумний дім”, в якому можна зробити будь-що, використовуючи голосову команду.

Alexa Voice надає вам доступ до хмарних можливостей Alexa з підтримкою API апаратних наборів, програмних засобів та документації.

Найкращим є те, що регулярні оновлення Alexa приносять нові функції на ваш пристрій і додають підтримку зростаючому асортименту сумісних пристроїв розумного дому.

1.5.6 Google Cloud Speech API

Google Cloud Speech API є частиною інфраструктури Google Cloud.

Цей API підтримує понад 110 мов. Система підтримує налаштування у вигляді надання списку можливих слів, які слід розпізнати (ця річ особливо корисна, якщо ви хочете використовувати розпізнавання мови в деяких пристроях чи інших ситуаціях, коли список можливих слів обмежений). API може працювати як в пакетному, так і в режимі реального часу. Він стійкий до побічних шумів в аудіо. Для деяких мов доступний фільтр нецензурних слів. Система побудована за допомогою нейронних мереж і може бути удосконалена з часом.

Файли, які ви хочете обробити, можна безпосередньо завантажити використовуючи API або зберегти у хмарному сховищі Google.

Ціна гнучка. До 60 хвилин обробленого аудіо безкоштовно для кожного користувача. Якщо ви хочете обробити більше 60 хвилин, вам слід заплатити 0,006 USD за 15 секунд.

Google Speech Cloud API став найбільш використовуваним API у 2020 році. Він має три основні способи розпізнавання мовлення:

- Синхронне розпізнавання мови
- Асинхронне розпізнавання мови
- Потокове розпізнавання мовлення

Синхронне розпізнавання мови повертає розпізнаний текст для коротких аудіофайлів (менше ніж ~ 1 хвилина) як тільки текст обробляється повністю.
Цей метод розпізнавання є найшвидшим та найпростішим, але підходить лише для коротких аудіо.

Асинхронне розпізнавання мови розпізнає аудіо чи відео файл з використанням декількох потоків, що забезпечує більшу швидкість для великих файлів у порівнянні з синхронним.

Потокове розпізнавання обробляє аудіо чи відео файли у реальному часі.

Воно є доступним лише з попередньою оплатою сервісів Google та являється рішенням для більш комплексних задач.

Програма базується на методі розпізнавання мови в синхронному режимі.

Причина використання цього методу розпізнавання мовлення полягає в тому, що користувач буде керувати комп’ютером використовуючи короткі слова та фрази, а також процес обробки мови в текст повинен бути якомога швидше. Найкращий спосіб забезпечити цю функцію - це використання синхронного розпізнавання мови, метод розпізнавання, де результати повертаються негайно.

1.6 Голосові помічники для управління комп’ютером

Віртуальний асистент або інтелектуальний голосовий помічник — це програмний агент, що може надавати персональну інформацію, виконувати завдання та послуги для окремої особи. У список доступних послуг входять пошук, керування музикою та пристроями розумного будинку, робота зі списками задач та електронною поштою, взаємодія з календарем і так далі. Взаємодія з такими помічниками зазвичай відбувається за допомогою голосу або тексту. Іноді стосовно віртуальних помічників з текстовим, чат-інтерфейсом застосовують термін "чат-бот".

Станом на 2020-й рік, ринок віртуальних помічників перебуває у стані активного розвитку: пожвавлюється інтерес як зі сторони користувачів, так і зі сторони великих компаній. Як результат, функціональність асистентів постійно розширюється, віртуальні помічники входять до складу мобільних та десктопних операційних систем, а на ринок виходять окремі "розумні" пристрої з акцентом на голосову взаємодію.

Найбільша користувацька база наразі належить продуктам від Apple, Google, Microsoft та Amazon. [10]

1.6.1 Siri

Siri - це вбудований персональний голосовий помічник, доступний для користувачів Apple.

Siri створена для того, щоб запропонувати вам безперебійний спосіб взаємодії з вашим iPhone, iPad, iPod Touch, Apple Watch, HomePod або Mac, коли ви розмовляєте з нею, щоб знайти або зробити те, що вам потрібно. Ви можете задати їй запитання, попросити показати вам щось, або дати їй команди, щоб вона виконувала їх від вашого імені.

Siri має доступ до будь-якого іншого вбудованого додатку на вашому пристрої Apple - пошти, контактів, повідомлень, карт, веб-браузеру тощо - і має доступ до баз даних кожної з цих програм. Врешті-решт, Сірі робить всю роботу за вас. [11]

Вона досить розумна, і часом навіть може пожартувати. Якщо необхідно, вона може прочитати ваш останній електронний лист, та надіслати повідомлення вашому другу, що ви запізнюєтеся; перетасувати порядок програвання музики в Apple Music; дасть вам знати, які фільми сьогодні показують в кінотеатрі, знайде стіл для трьох в Лондоні або зателефонує вашому другу; порекомендує хороший ресторан неподалік; установить будильник;

Це ще не все, насправді це, мабуть, навіть не половина.

Siri базується на галузях штучного інтелекту та обробки природних мов, і складається з трьох компонентів - розмовного інтерфейсу, особистого усвідомлення контексту та делегування послуг.

На даний момент є підтримка таких мов: іспанська, французька, німецька, італійська, японська, корейська, мандаринська, норвезька, кантонська, шведська, датська, голландська, російська, турецька , тайська та португальська.

Єдиним недоліком є те, що вона трохи повільно реагує, коли вам потрібно працювати з сторонніми додатками.

Тож ми бачимо, що Siri досить непоганий голосовий помічник, але він доступний лише для користувачів техніки Apple.

1.6.2 Cortana

Cortana - голосовий помічник від Microsoft. Спочатку він був розроблений для Windows Phone 8.1 і включений у Windows 10.

Як і голосовий помічник Apple - Siri, Cortana може виконувати різноманітні організаційні завдання для кінцевих користувачів, включаючи налаштування нагадувань, планування подій календаря, обчислення математичних задач та перетворення вимірювань та грошей.

Cortana має власний API і може працювати з різними універсальними додатками для Windows, а також сторонніми додатками, такими як Facebook та Twitter. Крім того, адміністратори можуть використовувати API, щоб налаштувати свої додатки для бізнесу або власні програми для взаємодії з Cortana. Cortana названа на честь персонажа жіночого штучного інтелекту у серії відеоігор Microsoft Halo.

Microsoft постачає оновлення Cortana незалежно від оновлень Windows 10, тому цифровий помічник може отримувати оновлення частіше.

На комп'ютерах з Windows 10 Cortana може відкривати програми, знаходити файли та читати чи надсилати електронні повідомлення. Користувачі можуть або набрати запит до Cortana, або увімкнути мікрофон та поговорити з програмою. Кортана інтегрується з новим веб-браузером Microsoft - Microsoft Edge та Internet Explorer (IE). В Edge користувачі можуть просто виділити слово чи фразу, і Cortana відображає більше інформації по темі. [12]

Cortana зберігає налаштування користувачів у зоні зберігання під назвою Notebook. Блокнот записує інтереси користувачів, улюблені місця та інформацію, яку вони хочуть відстежувати, включаючи їх улюблені спортивні команди та новини, які їх цікавлять. Для доступу до Notebook, користувачі просто відкривають Cortana та натискають Notebook.

Як і Siri прив'язана до техніки Apple, так і Cortana прив'язана до операційної системи Windows.

1.6.3 Google personal assistant

Google Assistant - голосовий помічник Google, який є покращенною версією Google Now. Спочатку Google Assistant спритно витягнув для вас відповідну інформацію.

Він знав, де ви працювали, ваші зустрічі та плани подорожей, спортивні команди, які вам сподобалися, і що вас зацікавило, щоб він міг представити вам важливу інформацію.

Google давно вбив Google Now, але Assistant живе в тому ж просторі, з'єднуючи ці персоналізовані елементи з широким діапазоном голосового управління. Google Assistant підтримує введення тексту або голосу, і він буде дотримуватися розмови незалежно від методу введення, який ви використовуєте.

Google Assistant пропонує голосові команди, голосовий пошук та керування голосом, дозволяючи виконувати ряд завдань після того, як ви сказали слова "Okay, Google" або "Hello, Google". Він розроблений, щоб надати вам розмовні взаємодії.

Google Assistant спочатку запускався на смартфонах Google Pixel та Google Home, але тепер він доступний майже для всіх сучасних пристроїв Android, включаючи пристрої Wear OS, Android TV і Nvidia Shield, а також у деяких автомобілях, якщо вони пропонують підтримку Android Auto.

Google Assistant є власником розумних динаміків Google Home, але він також широко доступний для інших смарт-динаміків від сторонніх виробників, включаючи Sony, Sonos, LG та Panasonic.

Розумні пристрої для дому, такі як Philips Hue, Nest продукти та асортимент Ikea Home Smart, можуть, наприклад, контролювати Google Assistant, і не лише через Google Home, а де б ви не мали взаємодії з Assistant.

Google розширив свою службу Google Assistant у 2017 році, щоб вона була доступна на інших мобільних пристроях. Це побачило впровадження програми Assistant для більшості телефонів Android. По суті, якщо у вас є Android, ваш телефон має Google Assistant, тому база користувачів для Google Assistant величезна.

Google Assistant може відповісти вам навіть тоді, коли ваш телефон Android заблокований, якщо ви ввімкнете це в своїх налаштуваннях.

Google Assistant також доступний на iPhone, хоча є деякі обмеження. [13]

Отже, Google Assistant - це зручний додаток для смартфонів на базі Android.

Висновки до розділу 1

У цьому розділі було розглянуто автоматичне розпізнавання мови, методи розпізнавання мови, а також було проведене порівняння існуючих API для перетворення мовлення в текст. Для даного дипломного проекту використовується Google Cloud Speech API, тому що ця система підтримує налаштування списку можливих слів, можна додавати свої слова та відповідні дії які буде виконувати комп'ютер. Також для API було обрано синхронне розпізнавання мови, яке являється найшвидшим тай найбільш надійним для коротких слів та фраз. Метою дипломного проекту є створення програмного забезпечення з зручним графічним інтерфейсом для виконання голосових команд, з можливістю додавати свої слова і відповідні дії які буде виконувати комп'ютер. Також, розглянувши найбільш популярні голосові помічники, ми побачили, що для Linux дистрибутивів немає аналогів.

РОЗДІЛ 2
ТЕХНІЧНІ ХАРАКТЕРИСТИКИ

2.1 Постановка задачі

2.1.1 Характеристика об’єкта автоматизації

Об’єктом автоматизації є процес виконання вказаних голосових команд.

На даний момент багато людей використовують голосових асистентів, тому що це досить швидкий варіант - використовувати голос для пошуку інформації, чи то для запуску програм.

Тому є потреба для створення голосового асистента, який би:

- здійснював пошук програм на комп’ютері та завантажував їх;
- здійснював пошук в інтернеті;
- завантажував сайти, які необхідні користувачу;
- мав змогу закривати не потрібні програми

Цей список можна продовжувати ще довго, за замовчуванням у програмі понад 15 можливих операцій, та є можливість їх додавати якщо потрібно щось власне.

Сам процес буде складатись з перегляду доступних команд запиту для даної програми, яка в свою чергу зможе відповісти на даний запит та виконати його.

Голосовий асистент зможе задовольнити на практиці і особисті вимоги користувачів, тому що код є досить гнучким, і для редагування, або додавання нових функцій буде потрібно мінімум знань з програмування. Завдяки методу декомпозиції розширення системи є не складною задачею .

Великі компанії, що пропонують своїх голосових помічників (Google, Apple, та інші), маючи список команд в кілька десятків тисяч, які постійно поповнюються, не можуть публікувати повний список всіх команд наявних у програмі. Тому в програму була введена “Довідка”. При використанні програми слід враховувати, що гарантується наявність тільки відповідних команд даного запиту, а не конкретних бажань користувача.

Знайти потрібну команду можна як безпосередньо за допомогою кнопки “Довідка”, так і через папку де знаходиться програма.

Так само буде реалізовано в даній програмі.

2.1.2 Вимоги до програми

З попереднього опису предметної області випливає, що даний дипломний проект потребує розробки програмного продукту, який дозволить спростити процес пошуку необхідної інформації та програми на ПК. Також продукт, що планується розробити, повинен охоплювати такі функції, як збереження введених команд та вивід здійснених.

Користувачем є клієнт, який має доступ до всіх команд, він вводить голосовий запит для пошуку пошуку. Створення програми, що автоматизує його діяльність є актуальною і необхідною задачею в сучасних умовах.

Програма передбачає розробку глосарію (далі – глосарій) для зберігання ключових слів. Також надання зручного інтерфейсу користувачеві.

Глосарій повинний містити наступні часто вирази, які часто використовуються: Відкрити, запустити, закрити, пошук та і.н.ш

Інтерфейс повинен бути зручним і комфортним, та не створювати проблем користувачеві. Програма повинна володіти всіма якостями комфортного людино-машинного інтерфейсу, як то: дружність, природність, узгодженість, не надмірність та повинен бути інтуїтивно зрозумілим.

2.2 Опис математичного методу рішення задачі

Загалом проектування повинно відбуватися методом декомпозиції, тобто розбиттям однієї великої задачі на декілька більш простих, кожна з яких також розбивається на підзадачі. Глибина декомпозиції, тобто глибина розбиття, визначається об’ємом величини задачі, яка утвориться після розбиття на основні розділи. Атомарною (неподільною) для проектування вважати ту задачу, яка повністю виконує одну логічно зв’язану дію.

Декомпозицією називається процес розбиття програми на частини. Легко зрозуміти, для чого необхідно проводити коректну декомпозицію: адже до вирішення об'ємної і складної задачі, залучається велика кількість людей, причому ця кількість зростає пропорційно складності розв'язуваної задачі. Цілком може статися, що регулярні контакти між цими людьми будуть деякою проблемою. Основне завдання необхідно розбити на частини, які можуть бути розроблені окремо, і які згодом можна було б легко об'єднати в одну систему. Подібне розділення дозволить займатися різними частинами проекту незалежно один від одного. Та що найважливіше – дуже спростить майбутній процес відладки.

За допомогою шаблону схеми класу можна створити новий клас використовуючи (ООП).

Об'єктно-орієнтоване програмування (ООП) - одна з парадигм програмування, яка розглядає програму як множину «об'єктів», що взаємодіють між собою. В ній використано декілька технологій від попередніх парадигм, зокрема успадкуванням, модульність, поліморфізм та інкапсуляцію. [20]

Призначенням глосарію тільки збереження окремих, не пов'язаних між собою виразів, його структура дуже проста. Проте одне з основних вимог до організації глосарію - це забезпечення можливості пошуку за значенням. В реальних глосаріях нерідко утримуються сотні або навіть тисячі виразів, теоретично може бути більше мільярда виразів. Наявність такої множини виразів і визначає корисність.

2.3 Структура вхідних даних

Вхідні дані формуються на основі первинних документів, підлягають обробці через відповідні форми, слугують базою для отримання вихідних даних.

Структуру вхідних даних проекту наведено у таблиці 2.1

Таблиця 2.1 - Структура вхідних даних

№ з/п	Назва	Тип	Розмір поля	Пояснення
	Statement	str	255	Запит

2.4 Структура вихідних даних

Вихідні дані це виконання програми та формування звітів виконаної команди. Структуру вхідних даних проекту наведено у таблиці 2.2

Таблиця 2.2 - Структура вихідних даних

№ з/п	Назва	Тип	Розмір поля	Пояснення
	url	str	255	Запит

2.5 Отримання даних з звукових сигналів

Порівнявши аналіз існуючих API для розпізнавання мови, було виявлено, що найпростішим та найефективнішим варіантом буде використовувати Google Cloud Speech API (рис 2.1).

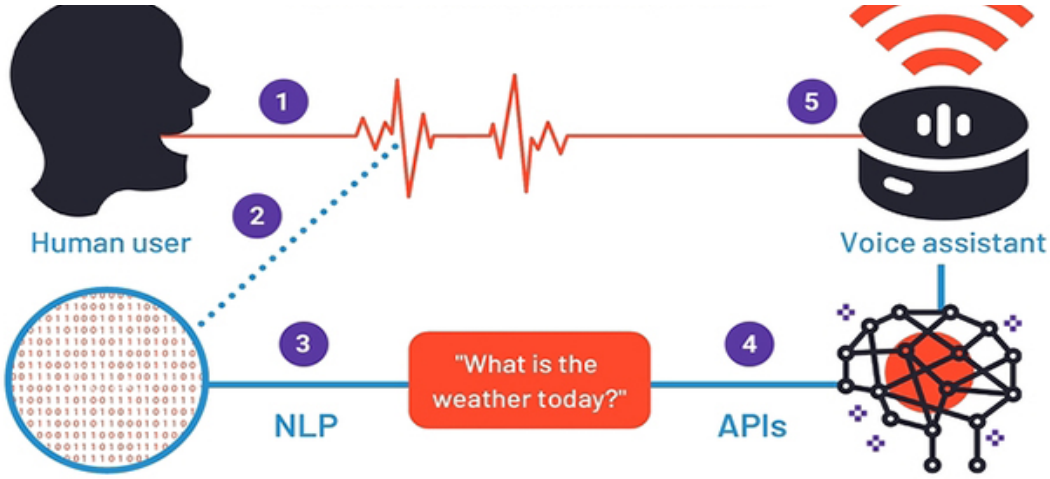


Рис.2.1 Проста схема розпізнавання мови за допомогою Google Cloud Speech API

Висновки до розділу 2

Програма «Голосове управління комп’ютером на основі глосарію за допомого алгоритмів розпізнання мови» представляє з себе програмний додаток з графічною оболонкою, перш за все для людей з обмеженими можливостями, але завдяки зручному інтерфейсу та простоті, являється зручним застосунком для будь-кого. В цьому розділі було проведено аналіз існуючих методів розпізнавання мови людини, та були розглянуті основні типи задач, які не можуть існувати без систем розпізнавання. Також було визначено необхідні вимоги та опис математичного рішення для програми.

РОЗДІЛ 3
КОМП'ЮТЕРНА СИСТЕМА

3.1 Технічні характеристики комп'ютера та зовнішніх пристроїв

Розробка даного програмного забезпечення проводилась на комп'ютері з даними характеристиками: персональний комп'ютер, об'єм оперативної пам'яті – 16 Гбайт, швидкість процесору – 2.6 ГГц, 6 ядер та 12 потоків, монітор з розширенням 2880x1800.

Для роботи розробленого програмного забезпечення повинні використовуватися наступні технічні засоби:

Персональний комп'ютер (системний блок, монітор, клавіатура, мишка, мікрофон).

Характеристики:

- Процессор 1.4 ГГц.
- Оперативна пам'ять 512 Мбайт.
- Кількість доступної пам'яті на накопичувачі більше 500МБ.
- Операційна система на ядрі Linux.
- Звукова карта
- Мікрофон

Рекомендовані вимоги до комп'ютера:

- Процессор 1.5 ГГц і більше.
- Оперативна пам'ять – 2056 Мбайт і більше.
- Кількість доступної пам'яті на накопичувачі більше 1ГБ
- Операційна система на ядрі Linux
- Звукова карта.
- Мікрофон.

3.2 Вибір програмних засобів та операційної системи

Усі елементи керування розробляються власноруч. Передача даних між Клієнтом та сервісом є важким організаційним процесом.

Але є велика кількість значущих переваг такого методу:

− легкість розбиття на модулі та програмування кожного з них окремо один він одного;

− з особливостей вище витікає легкість процесу відладки;

− дуже велика гнучкість та легкість внесення змін як простих, так і об’ємних;

− незалежність виконання від встановленої операційної системи;

− легкість проектування програми під особисті потреби користувача.

3.2.1 Мова програмування

Для реалізації задачі було обрано мову програмування Python.

Ідея Python виникла в 1989 році, коли її творець Гвідо ван Россум зіткнувся з недоліками мови ABC (а саме з розширюваністю). Россум розпочав роботу над розробкою нової мови, яка інтегрувала всі корисні функції мови ABC та нові бажані функції, такі як розширюваність та обробка виключень. Python 1.0 вийшов у 1994 році; вона запозичила модульну систему від Modula-3, мала можливість взаємодіяти з операційною системою Amoeba і включала функціональні засоби програмування. [1]

У 2000 році основна команда розробників Python перейшла на Beopen.com, а в жовтні 2000 року Python 2.0 був випущений з багатьма імпровізаціями, включаючи сміттєзбірник та підтримку Unicode.

У грудні 2008 року було випущено Python 3.0, відмовившись від зворотної сумісності та нового дизайну, щоб уникнути дублювання конструкцій та модулів. Це все ще багатопарадигмальна мова, яка пропонує розробникам параметри об'єктно-орієнтованого, структурованого програмування та функціонального програмування.

Сьогодні Python має декілька реалізацій, включаючи Jython, написаний мовою Java для віртуальної машини Java; IronPython написаний на C # для загальної мовної інфраструктури, а версія PyPy написана в RPython та переведена на C. Хоча ці реалізації працюють на рідній мові, на якій вони написані, вони також можуть взаємодіяти з іншими мовами за допомогою модулів. Більшість із цих модулів працюють за моделлю розвитку громади та є відкритими та безкоштовними.

Еталонної реалізацією Python є інтерпретатор CPython, що підтримує більшість активно використовуваних платформ . Він поширюється під вільною ліцензією Python Software Foundation License, що дозволяє використовувати його без обмежень в будь-яких додатках. Є реалізація інтерпретатора для JVM з можливістю компіляції, CLR, LLVM, інші незалежні реалізації. Проект PyPy використовує JIT-компіляцію, яка значно збільшує швидкість виконання Python-програм. [14]

Компанії, що розробляють програмне забезпечення, віддають перевагу мові Python через його універсальні можливості та меншу кількість програмних кодів. Майже 14% програмістів використовують його в таких операційних системах, як UNIX, Linux, Windows та Mac OS. Програмісти великих компаній використовують Python, оскільки він створив собі позначку в розробці програмного забезпечення з характерними функціями, такими як:

- Інтерактивність
- Інтерпретованість
- Модулярність
- Динамічність
- Об’єктно-орієнтований стиль
- Портатиність
- Високорівневість
- Розширення з C та C++

Також, ця мова надає великі стандартні бібліотеки. Більшість широко використовуваних завдань програмування вже написані в цих бібліотеках, що збільшує компактність коду.

Більш того, поріг входження низький, а код багато в чому лаконічний і зрозумілий навіть тому, хто ніколи його не використовував. За рахунок простоти коду подальший супровід програм, написаних на Python, стає легше і приємніше в порівнянні з Java або C ++. З точки зору бізнесу перевага полягає у скороченні витрат і збільшенні продуктивності праці співробітників [15]

Крім сотні тисяч індивідуальних розробників і невеликих компаній, Python підтримують такі гіганти IT як:

- Facebook
- Yandex
- RedHat
- Google
- Dropbox
- Telegram
- Mozilla
- Amazon

І саме головне, компанія Google надає дуже зручні API клієнти для Python.

Підвівши підсумки, маємо, що Python - це високорівнева мова програмування загального призначення, орієнтована на підвищення продуктивності розробника та читання коду; синтаксис ядра Python мінімалістичний; це мова програмування, що спеціально розроблена для написання швидкодіючих програм; є підтримка структурного, об'єктно-орієнтованого, функціонального, імперативного і аспектно-орієнтованого програмування; основні архітектурні риси - динамічна типізація, автоматичне керування пам'яттю, повна інтроспекція, механізм обробки виключень, підтримка багатопотокових обчислень, високорівневі структури даних; підтримується розбиття програм на модулі, які, в свою чергу, можуть об'єднуватися в пакети.

3.2.2 Інтегроване середовище розробки

Інтегроване середовище розробки - це програмне забезпечення, що забезпечує користувальницький інтерфейс для розробки коду, тестування та налагодження функцій. Інтегроване середовище розробки надає декілька інструментів та функцій для полегшення розробки та стандартизації на основі мови програмування, на якій розробник пише програмний код. Також є функції компіляції та інтерпретації програми. Деякі з широко використовуваних ідентифікаторів - це Eclipse для розробки програм програмування Java, Microsoft Visual Studio, Android Studio для розробки мобільних додатків, RStudio для програм R та PyCharm для програмування Python. [16]

За допомогою інтегрованого середовища розробки ви можете розробляти додатки, або динамічні веб-програми тощо. Воно включає редактор коду, компілятор або інтерпретатор та відладчик, щоб отримати доступ до графічного інтерфейсу користувача та дозволяє користувачеві писати та редагувати вихідний код у редактор коду.

Основна мета використання полягає в тому, щоб воно дозволило швидко та ефективно писати програмний код. Як вже було сказано вище, середовище включає вбудовані компілятори, які перетворюють програму в код машинного рівня або байт-код і економлять багато часу. Ви також можете вибрати кілька мов програмування на ваш вибір.

Хоча середовищ багато, та всі вони мають деякі спільні функції:

- Текстовий редактор
- Відладчик
- Компілятор або інтерпретатор
- Виконання написаного коду
- Підтримка хоча б однієї мови програмування, та всіх її версій
- Різноманітний набір плагінів

Написання голосового помічника здійснювалось в середовищі програмування JetBrains PyCharm Community Edition 2019. Середовище зображено на рис. 3.1

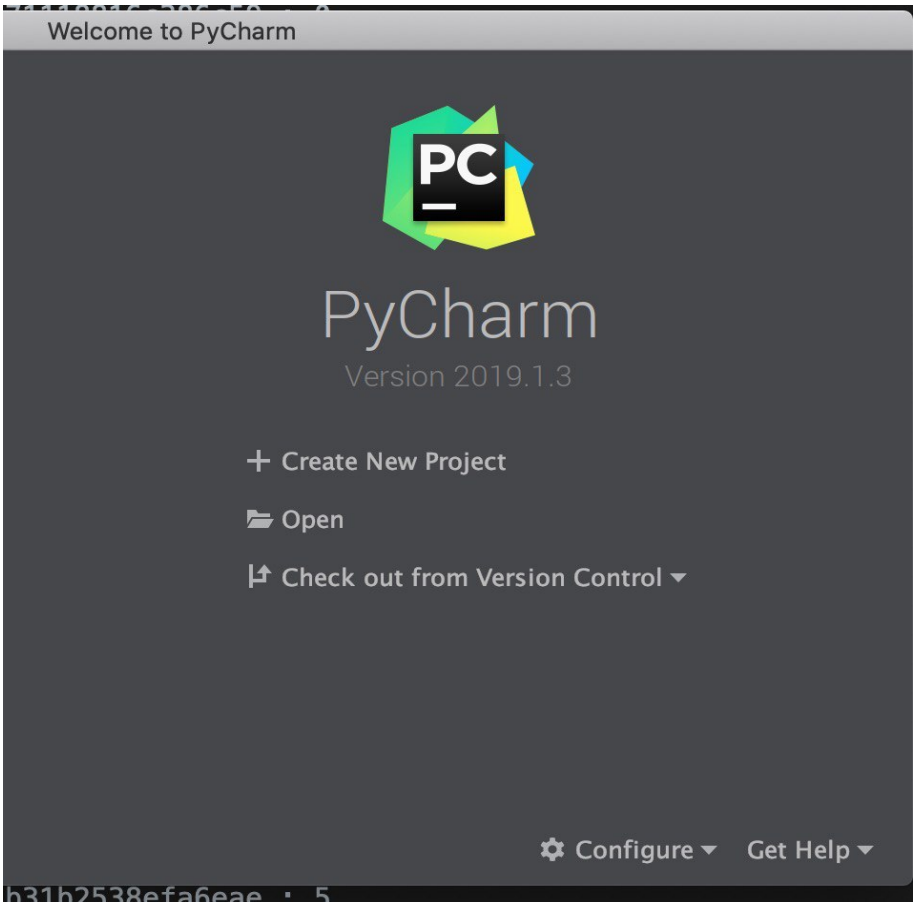


Рис. 3.1 JetBrains PyCharm Community Edition 2019

3.2.3 Користувацький інтерфейс

Для побудови користувацького інтерфейсу було використано бібліотеку PyQt5 для Python, що є оболочкою для фреймворку Qt написанного на C++.

Qt фреймворк вперше став загальнодоступним у травні 1995 року. Спочатку його розробили Хаавард Норд (генеральний директор Trolltech) та Ейрік Чамбе-Енг (президент Trolltech). [2]

Інтерес Хааварда до розробки графічного інтерфейсу C ++ розпочався в 1988 році, коли йому було доручено шведською компанією розробити графічний інтерфейс C++.

Через пару років, влітку 1990 року, Haavard та Eirik працювали разом над додатком бази даних C++ для ультразвукових зображень. Системі потрібно було працювати з графічним інтерфейсом на Unix, Macintosh та Windows. Одного разу того літа Хаавард та Ейрік вийшли на вулицю, щоб насолодитися сонячним світлом, і коли вони сиділи на лавці в парку, Хаавард сказав: «Нам потрібна об'єктно-орієнтована система відображення». В результаті дискусія заклала інтелектуальну основу для об'єктно-орієнтованого кросплатформенний графічний інтерфейс, який вони незабаром продовжуватимуть будувати.

20 травня 1995 року Qt 0,90 було завантажено на sunsite.unc.edu. Через шість днів про реліз було оголошено на comp.os.linux.announce. Це був перший публічний реліз Qt. Qt можна використовувати як для Windows, так і для Unix, пропонуючи однаковий API на обох платформах. З першого дня Qt був доступний за двома ліцензіями: для комерційної розробки була потрібна комерційна ліцензія, а для розробки з відкритим кодом - безкоштовна версія програмного забезпечення. Контракт Metis тримав Trolltech на плаву, тоді як протягом десяти довгих місяців ніхто не купував комерційну ліцензію Qt.

У березні 1996 року Європейське космічне агентство стало другим замовником Qt, придбавши десять комерційних ліцензій. З непохитною вірою Ерік і Хаавард найняли іншого розробника. Qt 0,97 було випущено наприкінці травня, а 24 вересня 1996 року вийшов Qt 1.0. До кінця року Qt досяг версії 1.1; вісім клієнтів, кожен в іншій країні, придбали між собою 18 ліцензій. Цього року також відбулося заснування проекту KDE, який очолив Маттіас Етріх.

Qt 1.2 було випущено в квітні 1997 року. Рішення Маттіаса Етріха використовувати Qt для створення KDE допомогло Qt стати фактичним стандартом для розробки графічного інтерфейсу C ++ в Linux. Qt 1.3 був випущений у вересні 1997 року.

Trolltech випустив Qtopia Core (тоді його називали Qt / Embedded) у 2000 році. Він був розроблений для роботи на вбудованих пристроях Linux та забезпечив власну віконну систему як легку заміну для X11.

Як Qt / X11, так і Qtopia Core тепер пропонувались під широко використовуваною загальною публічною ліцензією GNU (GNU), а також під комерційними ліцензіями. До кінця 2000 року Trolltech створила Trolltech Inc. (США) та випустила першу версію Qtopia, платформу додатків для мобільних телефонів та КПК. У 2001 та 2002 роках Qtopia Core виграв нагороду LinuxWorld за найкраще вбудоване рішення Linux, а в 2004 році таку ж відзнаку досяг Qtopia Phone.

Qt 3.0 був випущений у 2001 році. Тепер Qt був доступний у Windows, Mac OS X, Unix та Linux (настільний та вбудований). Qt 3 забезпечив 42 нові класи, і його код перевищив 500 000 рядків. Qt 3 був головним кроком вперед від Qt 2, включаючи значно вдосконалену підтримку локалі та Unicode, абсолютно новий віджет для перегляду та редагування тексту та регулярний клас вираження регулярних виразів. У 2002 році Qt 3 виграв Software Development Times „Jolt Productivity Award".

Також у 2005 році Trolltech відкрив представництво в Пекіні, щоб надати клієнтам у Китаї та регіоні послуги з продажу, навчання та технічну підтримку для Qtopia.

З моменту народження Trolltech популярність Qt зросла і продовжує зростати донині. Цей успіх є відображенням як якості Qt, так і того, наскільки приємно використовувати цей фреймворк. В останнє десятиліття Qt перейшло від продукту, яким користуються декілька вибраних «знаю», до того, яким щодня користуються тисячі клієнтів і десятки тисяч розробників з відкритим кодом у всьому світі. [17]

PyQt - це бібліотека, яка дозволяє використовувати Qt GUI на мові Python. Використовуючи його з Python, ви можете створювати додатки набагато швидше, не втрачаючи при цьому великої швидкості C ++.

PyQt5 практично повністю реалізує можливості Qt. А це понад 600 класів, більше 6000 функцій і методів. [3]

PyQt5 також включає в себе Qt Designer (Qt Creator) - дизайнер графічного інтерфейсу користувача. Програма pyuic генерує Python код з файлів, створених в Qt Designer.

Це робить PyQt5 дуже корисним інструментом для швидкого прототипування. Крім того, можна додавати нові графічні елементи управління, написані на Python, в Qt Designer (рис 3.2).

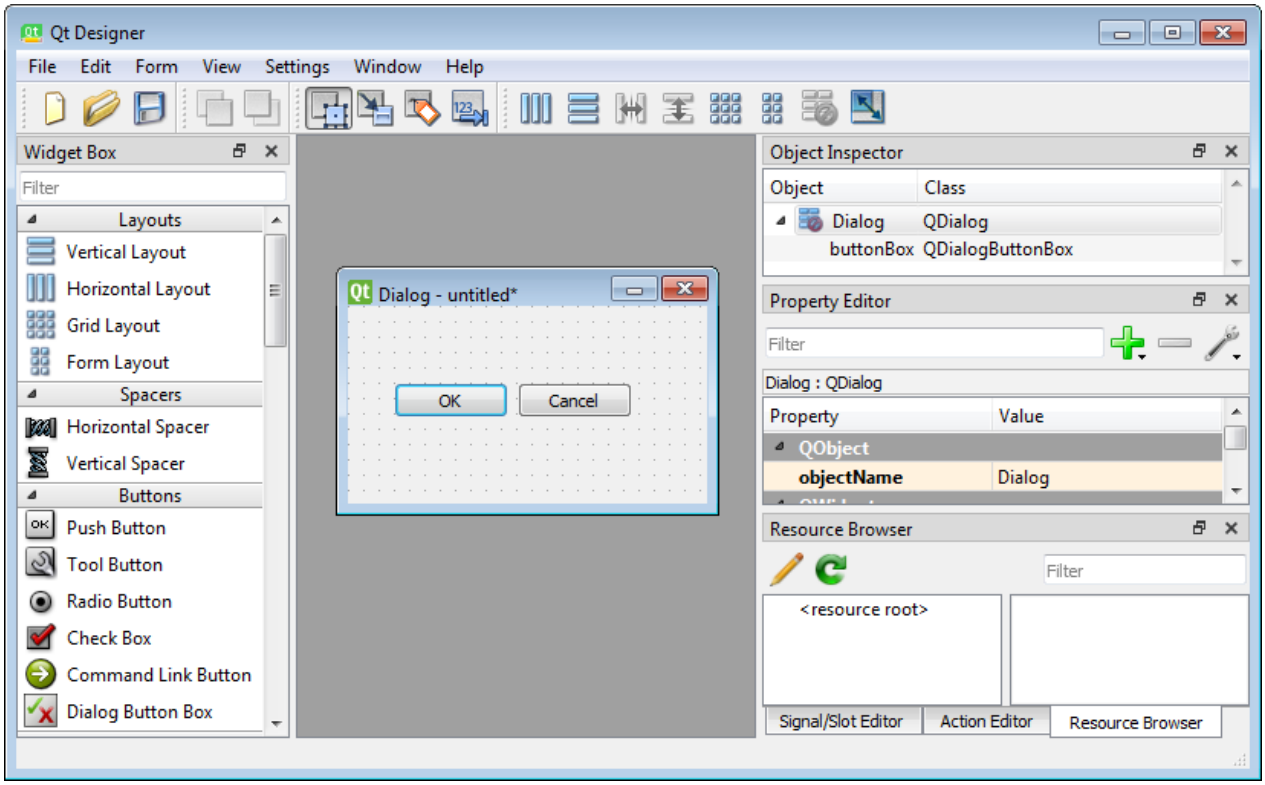


Рис. 3.2 Qt Designer

Раніше PyQt5 поставлявся разом із середовищем розробки Eric, яке написане на PyQt5. Eric має вбудований відладчик і може бути використана для створення консольних програм. Тепер вона доступна в якості окремого проекту .

Таким чином, основна перевага PyQt5 полягає в тому, що вона бере на себе всю роботу з графічною оболочкою, забезпечує швидкий доступ до класів, та функцій.

Іншим популярним засобом розробки GUI мовою Python є Tkinter. І Tkinter, і PyQt є корисними для проектування графічних інтерфейсів, але в той же час вони відрізняються з точки зору пристосованості та функціональності.

Здебільшого, Tkinter - про те, щоб самостійно писати графічний інтерфейс, запрограмувати налаштування чи функціональність у тому ж сценарії.

З іншого боку, в PyQt ви відокремлюєте графічний інтерфейс у сценарії та використовуєте свої знання Python від іншого сценарію. Тобто, як вже було сказано вище, в PyQt є засіб QtDesigner для створення дизайну, який потім можна інтегрувати у власну систму. В Tkinter такий засіб відсутній. Тож, у порівнянні з Tkinter плюсами використання PyQt є наступне:

- Гнучкість кодування - програмування GUI з Qt розроблено навколо концепції сигналів і слотів для встановлення зв'язку між об'єктами. Це дозволяє гнучкість під час роботи з подіями графічного інтерфейсу і призводить до більш гладкої кодової бази.
- Більше ніж фреймворк - Qt використовує широкий набір власних API платформ для створення мереж, створення бази даних та багато іншого. Він пропонує первинний доступ до них через унікальний API.
- Різні компоненти інтерфейсу користувача - Qt пропонує декілька віджетів, таких як кнопки або меню, розроблені з базовим виглядом на всіх підтримуваних платформах.
- Різні ресурси навчання - оскільки PyQt є одним з найбільш використовуваних фреймворків інтерфейсу для Python, ви можете отримати простий доступ до широкого набору документації.

Але, оскільки це в першу чергу фреймворк для C++, то з іншого боку він має і недоліки:

- Відсутність специфічної для Python документації для класів в PyQt5
- Це вимагає багато часу для розуміння всіх деталей PyQt, це означає, що це досить важка крива навчання

Переваги використання Tkinter:

- Доступно безкоштовно для комерційного використання.
- Він розміщений у базовій бібліотеці Python.
- Створення виконуваних файлів для додатків Tkinter є більш доступним, оскільки Tkinter включений в Python, і, як наслідок, він не має інших залежностей.

- Простий для розуміння та оволодіння, оскільки Tkinter - це обмежена бібліотека з простим API, що є основним вибором для швидкого створення простих графічних інтерфейсів для скриптів Python.

Та очевидні недоліки його використання:

- Tkinter не включає в себе складні віджети.
- Він не має подібного інструменту, як Qt Designer для Tkinter.
- Дизайн виглядає набагато гірше (рис 3.3)

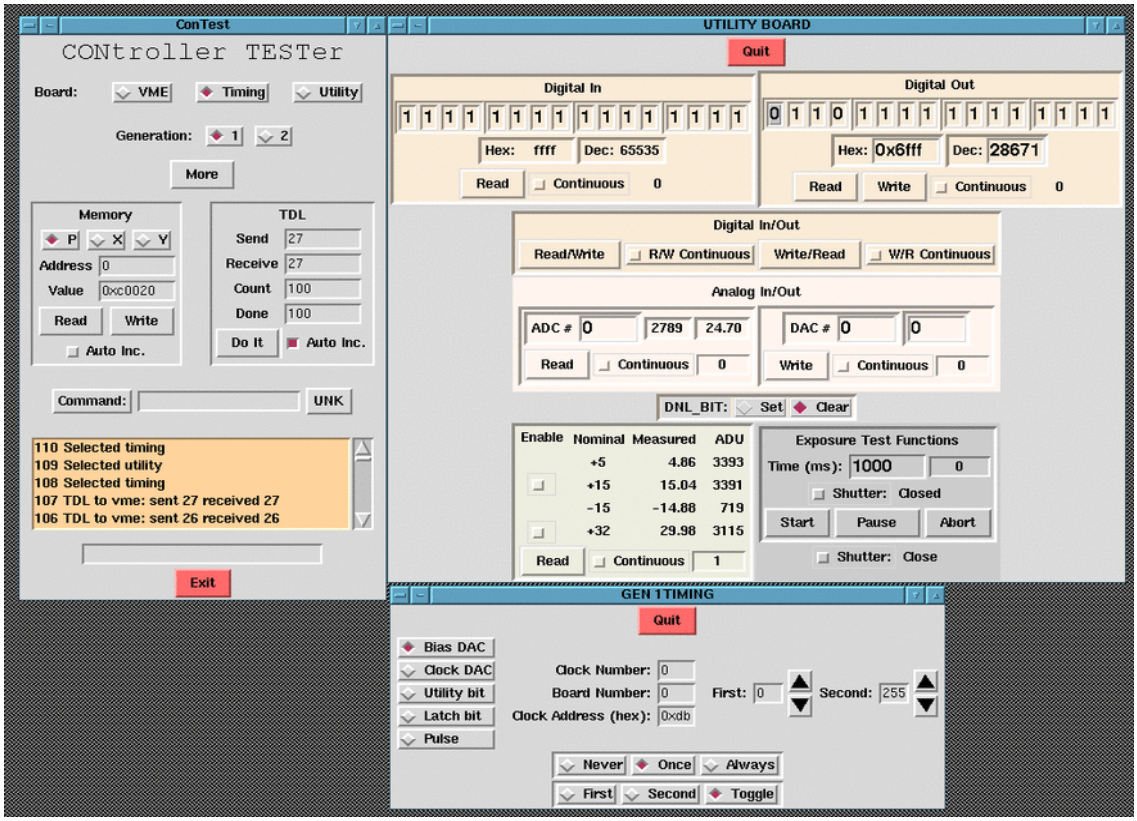


Рис. 3.3 Дизайн GUI з використанням Tkinter

Нарешті, останнім популярним засобом розробки GUI мовою Python є Kivy. Kivy - бібліотека Python, призначена для створення мультимедійних додатків, що підтримують мультимедіа. Мета - дозволити швидке та просте проектування взаємодії, а також швидке прототипування, багаторазовий код та функції розгортання.

Kivy розроблений на базі OpenGL, і його можна використовувати з декількома пристроями введення. Коли PyQt представляє собою повноцінний фреймворк для розробки графічного інтерфейсу, Kivy - більше використовується для міжплатформових засобів мобільної розробки.

Kivy працює безперебійно на Linux, Windows, OS X, Android та Raspberry Pi. І значна перевага використання Kivy os в тому, що ви можете запускати один і той же код на всіх підтримуваних платформах.

Все це досягається у вигляді дерева, який є дещо більш точним та організованим у порівнянні з іншими інструментами для програмування.

Оскільки Kivy більш орієнтований під мобільну розробку, в ньому також немає комплексних речей, складних віджетів тощо. Приклад додатку написаного за допомогою Kivy зображено на рис 3.4

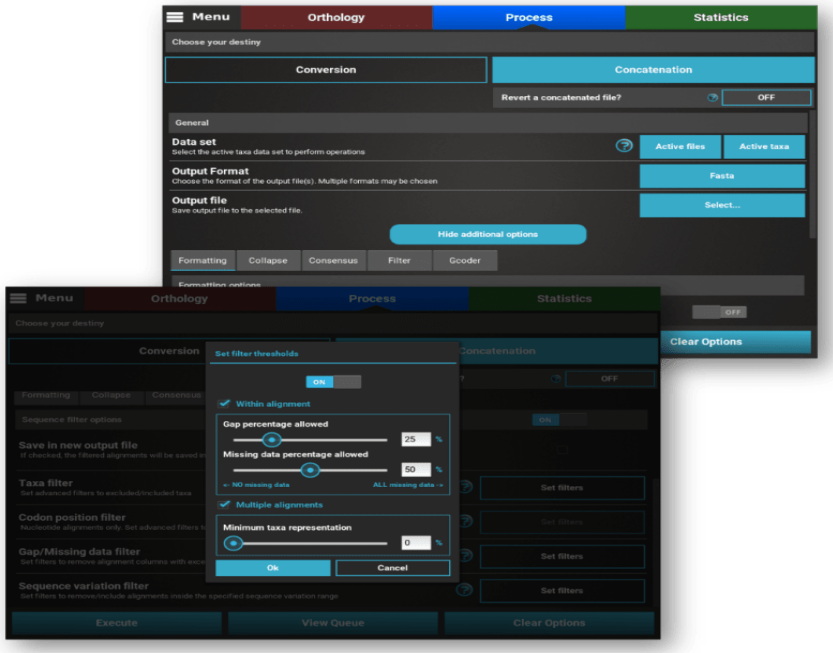


Рис. 3.4 Дизайн GUI з використанням Kivy

Отже, ми побачили що найкращим засобом для розробки графічного інтерфейсу є PyQt, так як в Tkinter неможливо зробити повноцінний комплексний додаток за відсутності складних віджетів, а Kivy більш орієнтований під мобільні додатки.

3.2.4 База даних

Для зберігання глосарію нам потрібна база даних. У цьому розділі порівнюються та протиставляються три найбільш відомі системи управління базами даних (СУБД): SQLite, MySQL та PostgreSQL. Зокрема, їх переваги недоліки.

У інструментах управління базами даних переважає модель реляційних даних, яка організовує дані в таблиці рядків і стовпців. Сьогодні існують інші моделі даних, включаючи NoSQL та NewSQL, але системи управління реляційними базами даних залишаються домінуючими для зберігання та управління даними по всьому світу.

Бази даних - це логічно модельовані кластери інформації або даних. З іншого боку, система управління базами даних (СУБД) - це комп'ютерна програма, яка взаємодіє з базою даних. СУБД дозволяє контролювати доступ до бази даних, записувати дані, виконувати запити та виконувати будь-які інші завдання, пов'язані з управлінням базами даних. Хоча системи управління базами даних часто називають "базами даних", ці два терміни не є взаємозамінними. Базою даних може бути будь-яка колекція даних, а не лише одна, що зберігається на комп'ютері, тоді як СУБД - це програмне забезпечення, яке дозволяє взаємодіяти з базою даних.

Усі системи управління базами даних мають базову модель, яка структурує спосіб зберігання та доступу до даних. Система управління реляційними базами даних - це СУБД, в якій використовується модель реляційних даних. У цій моделі дані впорядковуються в таблиці, які в контексті реляційних систем баз данних більш офіційно називаються відносинами. Відношення - це набір кортежів або рядків у таблиці, де кожен кортеж розділяє набір атрибутів або стовпців.

Більшість реляційних баз даних використовують структуровану мову запитів (SQL) для управління та запиту даних. Однак багато СУБД використовують свій власний діалект SQL, який може мати певні обмеження чи розширення. Ці розширення, як правило, містять додаткові функції, які дозволяють користувачам виконувати більш складні операції, ніж вони могли б зі стандартним SQL.

Кожному стовпцю присвоюється тип даних, який диктує, які записи дозволяються в цьому стовпці.

Різні СУБД реалізують різні типи даних, які не завжди безпосередньо взаємозамінні. Деякі типові дані включають дати, рядки, цілі числа та булеві значення.

Числові типи даних можуть бути зі знаком, тобто вони можуть представляти як позитивні, так і негативні числа, або без знаку, а значить, вони можуть представляти лише додатні числа. Наприклад, крихітний тип даних MySQL може містити 8 біт даних, що дорівнює 256 можливим значенням. Діапазон даних зі знаком цього типу становить від -128 до 127, тоді як діапазон без знаку - від 0 до 255. [21]

Іноді адміністратор бази даних накладає обмеження на таблицю, щоб обмежити, які значення можуть бути введені в неї. Обмеження зазвичай стосується одного конкретного стовпця, але деякі обмеження можуть застосовуватися і до цілої таблиці. Ось деякі обмеження, які зазвичай використовуються в SQL:

- Unique: Застосування цього обмеження до стовпця гарантує, що жодна з двох записів у цьому стовпці не однакова.
- Not Null: Це обмеження гарантує, що у стовпці немає записів NULL.
- Primary Key Комбінація UNIQUE та NOT NULL, обмеження PRIMARY KEY забезпечує відсутність запису в стовпці NULL та кожне введення є єдиним на всю таблицю.
- Foreign Key: Foreign Key - це стовпець в одній таблиці, який посилається на Primary Key іншої таблиці. Це обмеження використовується для з'єднання двох таблиць разом: записи до стовпця Foreign Key повинні вже існувати в батьківському стовпчику Primary Key, щоб процес запису був успішним.
- Check: Це обмеження обмежує діапазон значень, який можна ввести у стовпчик. Наприклад, якщо ваша програма призначена лише для жителів Аляски, ви можете додати обмеження Check в стовпчик з поштовим індексом, щоб дозволити записи лише між 99501 і 99950.

- Default: Це забезпечує значення за замовчуванням для даного стовпця. Якщо не вказано інше значення, SQL автоматично вводить значення за замовчуванням.
- Index: Використовується для швидшого отримання даних із таблиці, це обмеження схоже на індекс у підручнику: замість того, щоб переглядати кожен запис у таблиці, запит повинен лише переглянути записи з індексованого стовпця, щоб знайти потрібні результати.

Тепер, коли ми загалом розглянули базові знання з систем управління реляційними базами даних, давайте перейдемо до першої з трьох реляційних баз даних із відкритим кодом: SQLite.

SQLite - це автономна, заснована на файлах та повністю відкрита RDBMS, відома своєю портативністю, надійністю та високою продуктивністю навіть у середовищах з низькою пам’яттю. Його транзакції відповідають сумісності, навіть у випадках, коли система виходить з ладу або перебуває в роботі відключення електроенергії.

Веб-сайт проекту SQLite описує його як базу даних "без сервера". Більшість двигунів реляційних баз даних реалізовані як серверний процес, в якому програми спілкуються з хост-сервером за допомогою міжпроцесорної комунікації, яка ретранслює запити. Однак, якщо використовувати SQLite, будь-який процес, який отримує доступ до бази даних, читає і записує в файл диска бази даних безпосередньо. Це спрощує процес налаштування SQLite, оскільки він виключає будь-яку потребу в налаштуванні серверного процесу. Крім того, для програм, які використовують базу даних SQLite, немає необхідної конфігурації: все, що їм потрібно, - це доступ до диска.

SQLite - це безкоштовне програмне забезпечення з відкритим кодом, і для його використання не потрібна спеціальна ліцензія. Однак проект пропонує декілька розширень - кожне за одноразову плату - які допомагають стискати та шифрувати дані. Крім того, проект пропонує різні комерційні пакети підтримки, кожен за щорічну плату.

Переваги SQLite:

- Бібліотека SQLite дуже легка. Хоча простір, який він використовує, змінюється залежно від системи, де він встановлений, він може займати менше 600 Кбіт місця. Крім того, він повністю автономний, це означає, що для роботи SQLite у вашій системі немає жодних зовнішніх залежностей.
- Зручний користування: SQLite іноді описується як база даних з нульовою конфігурацією, яка готова до використання “з-під коробки”. SQLite не запускається як серверний процес, а це означає, що його ніколи не потрібно зупиняти, запускати чи перезавантажувати і не має жодних файлів конфігурації, якими потрібно керувати. Ці функції допомагають впорядкувати шлях від встановлення SQLite до його інтеграції з додатком.
- Портативний: На відміну від інших систем управління базами даних, які зазвичай зберігають дані у великій партії окремих файлів, вся база даних SQLite зберігається в одному файлі. Цей файл може бути розташований у будь-якому місці ієрархії каталогів, і ним можна ділитися через знімний носій або протокол передачі файлів.

Недоліки SQLite:

- Обмежена паралельність: Хоча декілька процесів можуть отримати доступ до бази даних SQLite одночасно та запитувати їх, лише один процес може вносити зміни до бази даних в будь-який момент часу. Це означає, що SQLite підтримує більшу сумісність, ніж більшість інших вбудованих систем управління базами даних, але не стільки, скільки СУБД клієнтів / серверів, таких як MySQL або PostgreSQL.
- Відсутність управління користувачами: Системи баз даних часто підтримують користувачів або керують з'єднаннями з попередньо визначеними правами доступу до бази даних та таблиць.

Оскільки SQLite читає та записує безпосередньо у звичайний файл диска, єдині застосовні дозволи доступу є типовими дозволами доступу базової операційної системи. Це робить SQLite поганим вибором для програм, які потребують декількох користувачів із спеціальними дозволами доступу.

- Безпека: Двигун бази даних, який використовує сервер, може в деяких випадках забезпечити кращий захист від помилок у клієнтській програмі, ніж база даних без серверів, як SQLite. Наприклад, вхідні покажчики в клієнті не можуть пошкодити пам'ять на сервері. Крім того, оскільки сервер - це єдиний стійкий процес, база даних клієнт-сервер може контролювати доступ до даних з більшою точністю, ніж база даних без сервера, що дозволяє більш дрібно блокувати та покращувати сумісність.

Згідно рейтингу DB-Engines, MySQL була найпопулярнішою СУБД з відкритим кодом з моменту, коли сайт почав відслідковувати популярність бази даних у 2012 році. Це продукт, який використовують багато найбільших у світі веб-сайтів та компаній, включаючи Twitter, Facebook , Netflix та Spotify. Початок роботи з MySQL порівняно простий, завдяки великій частині його вичерпній документації та великій спільноті розробників, а також великій кількості ресурсів, пов’язаних з MySQL в Інтернеті.

MySQL був розроблений для швидкості та надійності за рахунок повного дотримання стандартів SQL. Він оснащений різними режимами та розширеннями SQL. На відміну від програм, що використовують SQLite, програми, що використовують базу даних MySQL, отримують доступ до нього через окремий процес демон. Оскільки серверний процес стоїть між базою даних та іншими програмами, він дозволяє забезпечити більший контроль над тим, хто має доступ до бази даних.

MySQL надихнув безліч сторонніх додатків, інструментів та інтегрованих бібліотек, які розширюють його функціональність та допомагають полегшити роботу.

Деякі з найбільш широко використовуваних цих сторонніх інструментів - phpMyAdmin, DBeaver та HeidiSQL.

PostgreSQL, також відомий як Postgres, вважається "найсучаснішою реляційною базою даних з відкритим кодом у світі". Він був створений з метою бути високорозширним (тобто для сайтів та додатків з великою нагрузкою) та відповідати стандартам. PostgreSQL - це об'єктно-реляційна база даних, тобто, хоча це переважно реляційна база даних, вона також включає такі функції, як спадкування таблиці та перевантаження функцій, які частіше асоціюються з об’єктними базами даних.

Postgres здатний ефективно впоратися з декількома завданнями одночасно, характеристикою, відомою як паралельність. Це досягається без блокування читання завдяки впровадженню мультиверсійного контролю за сумісністю, що забезпечує атомність, послідовність, ізоляцію та довговічність транзакцій, також відомих як відповідність ACID.

PostgreSQL не настільки широко використовується, як MySQL, але все ж існує ряд сторонніх інструментів і бібліотек, призначених для спрощення роботи з PostgreSQL, включаючи pgAdmin та Postbird.

Десятиліттями реляційна база даних (СУБД) була основою для більшості застосувань.

Однак за останнє десятиліття багато припущень, які зумовили розвиток попередніх реляційних баз даних, змінилися:

- Вимагає більш високої продуктивності розробників та швидшого виходу на ринок, завдяки традиційним жорстким моделям реляційних даних та розробці монолітних додатків, що поступаються місцем гнучким методологіям, мікросервісам та DevOps, стискаючи цикли випуску з місяців і років до днів і тижнів.
- Необхідність управління збільшенням нових типів даних, що швидко змінюються, - структурованих, напівструктурованих та поліморфних даних, породжених новими класами веб-, мобільних, соціальних додатків.

- Оптовий перехід до розподілених систем та хмарних обчислень, що дозволяє розробникам використовувати інфраструктуру на вимогу, з можливістю обслуговувати аудиторію в будь-якому місці, де вони працюють і грають по всьому світу, дотримуючись при цьому цілого нового набору регуляторних вимог за суверенітет даних.

Отже, виникли NoSQL бази даних, подібні до MongoDB, для обробки вимог нових програм та модернізації існуючих навантажень. Завдяки підтримці багатодокументних транзакцій ACID в MongoDB розробникам простіше, ніж будь-коли, створювати програми. [22]

MongoDB - це відкрита нетаблична база даних, розроблена компанією MongoDB, Inc. MongoDB зберігає дані як документи у двійковому представництві під назвою BSON. Пов'язана інформація зберігається разом в одному документі для швидкого доступу до запитів через мову запитів MongoDB.

Модель даних документа документа MongoDB природно відображає об'єкти в коді програми, що спрощує розробку та вивчення. Документи дають можливість представляти ієрархічні зв'язки для легкого зберігання масивів та інших більш складних структур. Рідні, ідіоматичні драйвери надаються для більше десятка мов - а спільнота створила ще десятки - що дозволяє спеціальні запити, агрегація в режимі реального часу та багата індексація, щоб забезпечити потужні програмні способи доступу та аналізу даних будь-якої структури.

Завдяки транзакціям з декількома документами MongoDB є єдиною базою даних, яка поєднує гарантії ACID традиційних реляційних баз даних, швидкості, гнучкості та потужності моделі документа, а також інтелектуальної розподіленої системи проектує масштаб і розміщує дані там, де це потрібно.

За допомогою ізоляції знімків, транзакції забезпечують послідовний перегляд даних та примушують виконувати все або нічого, щоб підтримувати цілісність даних. Операції в MongoDB є мульти-операторами, з подібним синтаксисом (наприклад, start_transaction і commit_transaction), і тому їх легко додати до будь-якої програми будь-кому, хто має попередній досвід транзакцій.

На відміну від PostgreSQL та інших реляційних баз даних, MongoDB побудований на архітектурі розподіленої системи, а не на монолітній конструкції одного вузла. Як результат, MongoDB пропонує нестандартне масштабування та локалізацію даних з набором реплік, щоб підтримувати доступність постійно.

Отже, розглянувши всі можливі варіант баз даних, доцільнішим вибором вважається SQLite, це швидкий і простий засіб для додатків без високої навантаженості. MySQL то PostgreSQL являються занадто комплексними для цього додатку, тож їх використання буде надлишком роботи. MongoDB також є непоганим вибором, але глосарій краще зберігати в класичній реляційній базі даних у структурованому вигляді.

3.2.5 Операційна система

Linux - це безкоштовна та відкрита операційна система, заснована на стандартах Unix. Вона надає інтерфейс програмування, а також інтерфейс користувача, сумісний із системами на базі Unix та надає велику кількість різноманітних програм. Система Linux також містить безліч окремо розроблених елементів, в результаті чого система Unix є повністю сумісною і не містить власного коду.

Традиційне монолітне ядро використовується в ядрі Linux з метою продуктивності, але його модульна функція дозволяє більшості драйверів динамічно завантажуватись під час виконання. Linux захищає процеси і є багатокористувацькою системою. Міжпроцесовий зв'язок підтримується обома механізмами, такими як черга повідомлень, спільна пам'ять та семафор.

Абстрактний шар використовується в Linux для управління різними файловими системами, але для користувачів файлова система виглядає як ієрархічне дерево каталогів. Вона також підтримує мережеві, орієнтовані на пристрої та віртуальні файлові системи. Доступ до дискового сховища здійснюється через кеш сторінки. Для мінімізації дублювання спільних даних між різними процесами система управління пам’яттю використовує спільний доступ до сторінок і копіювання під час запису.

Типовим користувальницьким інтерфейсом є GNOME, Ubuntu Unity (у старих версіях). [18]

Ключові відмінності між Linux та операційною системою Windows:

- Linux - це безкоштовна та відкрита операційна система, тоді як Windows - комерційна операційна система, вихідний код якої недоступний.
- Windows не налаштовується так гнучко відносно Linux, де користувач може змінювати і код ядра операційної системи і зовнішній вигляд.
- Linux забезпечує більший рівень безпеки, ніж Windows.
- Windows має завантажуватися з основного розділу. На відміну від цього в Linux немає такого обмеження, воно може бути завантажено з основного або логічного розділу.
- Linux використовує монолітне ядро тоді як Windows використовує мікроядро, ефективність роботи якого нижче.

Переваги операційної системи на ядрі Linux:

- Операційна система Linux – безкоштовна. Ви можете вибрати сподобавшийся вам дистрибутив та завантажити його абсолютно вільно;
- Програми для Linux безкоштовні. Вам не потрібно купувати кожну програму окремо, не потрібно шукати якісь кряки, серійні номера та інше.
- Дистрибутивів Linux дуже багато. Вибирайте будь-який під свої потреби. Під різні апаратні можливості, під різні завдання і цілі, для досвідчених користувачів та для новачків;
- Linux працює «з-під коробки». Деякі дистрибутиви встановлюються відразу з повним набором необхідних програм, кодеків, драйверів і інших необхідних компонентів;
- Всі програми можна встановити централізовано. Не потрібно заходити на десятки сайтів і завантажувати кожну програму окремо.

Як було сказано в першій частині цієї роботи, на даний момент немає безкоштовних аналогів Siri або Cortana для Linux дистрибутивів. Ціллю цієї роботи є також саме розробка голосового помічника під будь-який Linux дистрибутив. Тож у якості операційної системи використовується Mac OS X.

3.2.5 Інсталяція програмного додатку

Для полегшення інсталяції програмного додатку використовується PyInstaller.

PyInstaller поєднує додаток Python та всі його залежності в один пакет. Користувач може запустити додаток, не встановлюючи інтерпретатора Python та будь-яких модулів.

PyInstaller читає написаний вами сценарій Python. Він аналізує ваш код, щоб виявити будь-який інший модуль і бібліотеку, що потрібен вашому сценарію для виконання. Потім він збирає копії всіх цих файлів - включаючи активний інтерпретатор Python, і розміщує їх зі своїм сценарієм в одній папці або в одному виконаному файлі.

PyInstaller тестується на Windows, Mac OS X та GNU / Linux. Однак це не крос-компілятор: щоб зробити додаток для Windows, ви запускаєте PyInstaller в Windows; щоб створити додаток GNU / Linux, ви запускаєте його в GNU / Linux тощо. PyInstaller успішно використовується з AIX, Solaris, FreeBSD і OpenBSD. [19]

Основні переваги:

- Працює з будь-якою версією Python 2.7 / 3.x.
- Повністю мультиплатформна і використовує підтримку ОС для завантаження динамічних бібліотек, забезпечуючи таким чином повну сумісність.
- Правильно з’єднуються основні пакети Python, такі як numpy, PyQt4, PyQt5, PySide, Django, wxPython, matplotlib та інші, що не є в наявності.
- Сумісний із багатьма сторонніми пакетами, які не продаються. (Усі необхідні прийоми для роботи зовнішніх пакетів уже інтегровані.)

Висновки до розділу 3

У цьому розділі було визначено необхідні технічні характеристики комп’ютера та зовнішніх пристроїв для робота програми, а також були описані можливі бібліотеки та мови програмування для рішення задачі. Було встановлено, що даний проект доцільніше робити на мові програмування Python 3.x з використанням фреймворку PyQt5 для GUI, та з використанням Linux дистрибутиву. Для коректної роботи програми треба відслідковувати підключення до мережі інтернет. Після внесення суттєвих змін в структуру програми необхідно перекомпілювати проект та бажано протестувати модифікований додаток. Було порівняно можливі варіанти баз даних, та визначено що найкращим рішенням для даного додатку є SQLite.

РОЗДІЛ 4
ОБ’ЄКТНО-ОРІЄНТОВАНЕ ПРОЕКТУВАННЯ

4.1 Інтерфейс програми

Специфікації класів PyQT5 дозволяють побудувати зручний та комфортний інтерфейс.

Основні модулі які використовуються для цього:

- QtCore-основні не графічні класи: система сигналів та слотів, платформо незалежні, потоки, розподільчої пам'яті та регулярних виразів
- QtGui-компоненти графічного інтерфейсу (елементи управління), засновані на візуальному представленні.
- QtNetwork - класи для мережевого програмування. Наприклад, клієнтів і серверів через UDP і TCP.
- QtOpenGL - класи, що дозволяють використовувати OpenGL і 3D-графіку в додатках PyQt

Ліцензія PyQt5 продиктована досить складними відносинами в сфері застосування цієї надбудови (прив'язки) над графічною бібліотекою Qt. Відповідно до ліцензії GPL, можливе використання бібліотеки для створення програм з відкритим вихідним кодом. Якщо немає можливості розповсюджувати код відповідно до умов GPL, то можна придбати комерційну ліцензію .

З боку Nokia були спроби домовитися з Riverbank Computing на предмет зміни ліцензії, але вони не увінчалися успіхом. В результаті виник проект під назвою PySide - аналог PyQt5, в тому числі зберігає сумісність з останніми на рівні API, але випущений під ліцензією LGPL, щоб дати можливість розробникам і комерційних проектів безкоштовно використовувати Python в зв'язці з Qt.

Однак Riverbank не виключає можливості ліцензування PyQt5 під LGPL в майбутньому

Початкове вікно програми зображено на рис 3.1.

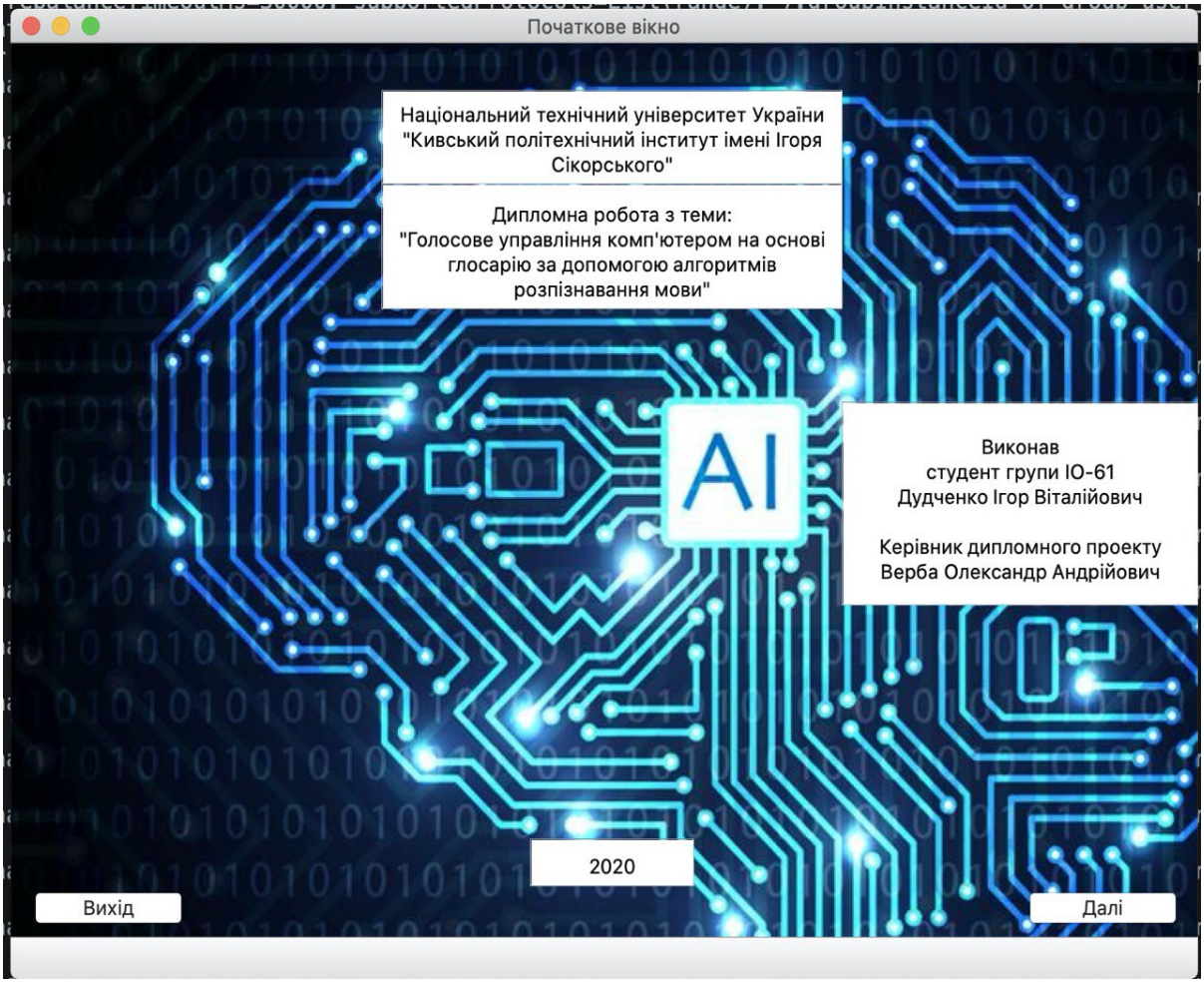


Рис. 3.1 - Стартова форма програми

Після натиснення кнопки “Далі” це вікно закривається, та відкривається головне вікно програми, яке розташоване в центрі екрану, та активне на протязі роботи програми, воно буде основним елементом взаємодії і дозволить зручно користуватися керуючими модулями рис. 3.2.

Рис. 3.2 – Головне вікно програми

У центральному вікні відображаються всі логи програми для зручності.

На сторінці “Довідка” надається опис функціоналу програми (рис 3.3). Ввід даних буде здійснюватися через натиснення на кнопку «Запуск», яка знаходиться на головному меню (рис 3.2) або за допомогою натиснення клавіши Ctrl. Для виконання дії потрібно буде дати голосовий запит.

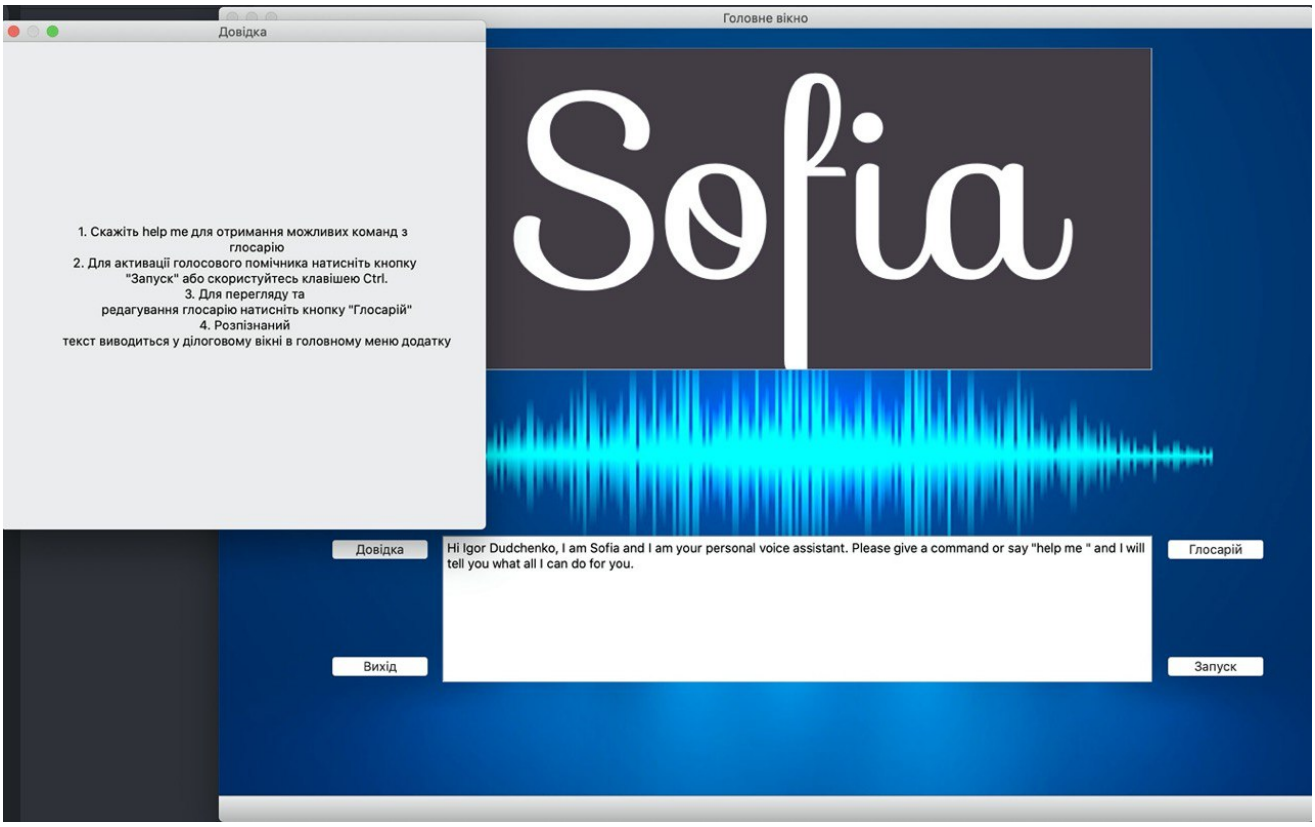


Рис. 3.3 – Вікно довідки

При натисненні кнопки “Глосарій” буде відкриватись вікно з даними з SQLite (рис 3.4). У цьому вікні можна редагувати існуючі ключові слова та модулі глосарію, а також додавати нові. Саме по цим ключовим словам запускається відповідний модуль програми.

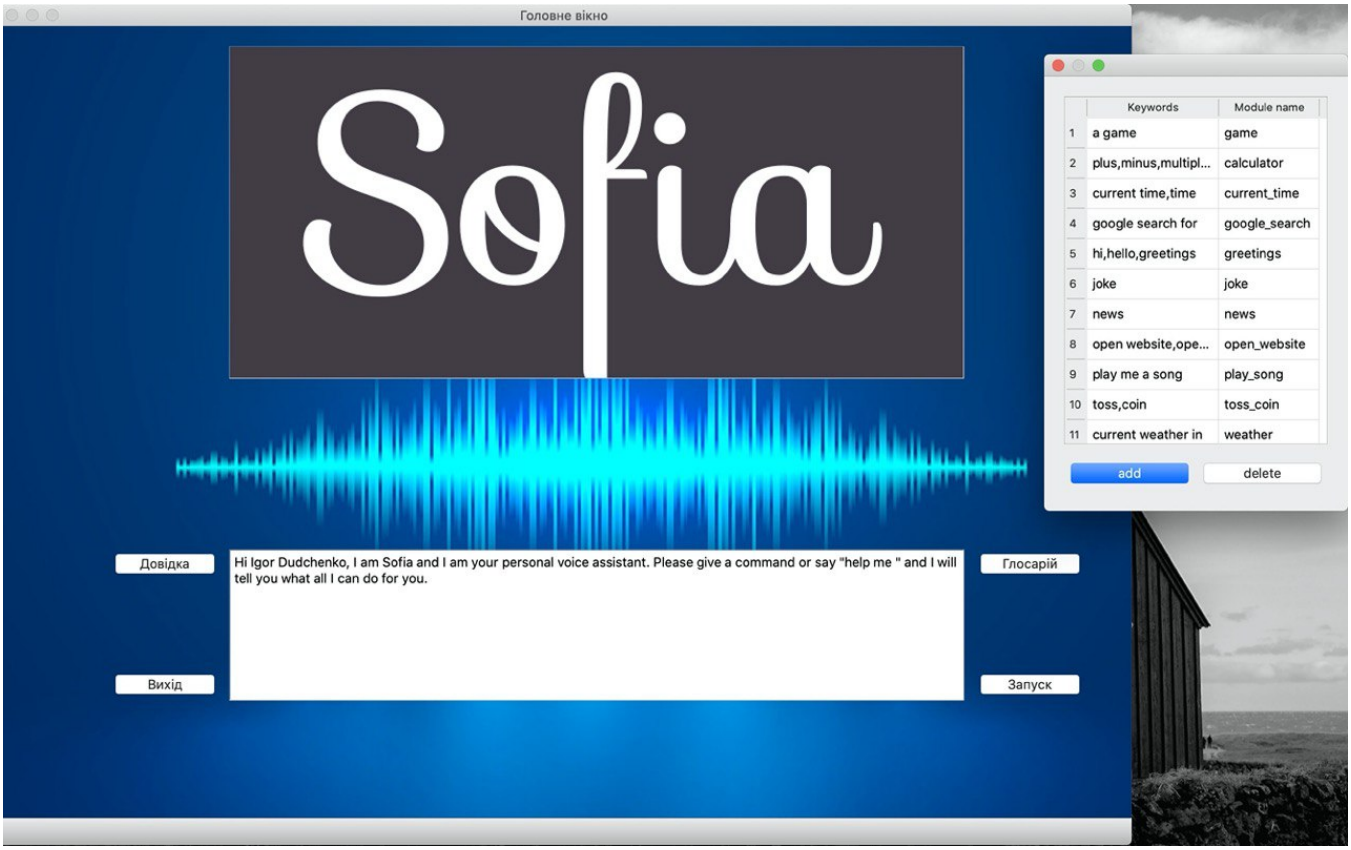


Рис. 3.4 – Вікно глосарію

4.2 Алгоритм рішення задачі

При завантаженні головної форми користувач має змогу вибрати необхідну кнопку та розпочати роботу.

Реалізація інтерфейсу програми виконана з використанням підходу програмування (ООП).

На відміну від традиційних поглядів, коли програму розглядали як набір підпрограм, або як перелік інструкцій комп'ютеру, ООП-програми можна вважати сукупністю об'єктів. Відповідно до парадигми об'єктно-орієнтованого програмування, кожний об'єкт здатний отримувати повідомлення, обробляти дані, та надсилати повідомлення іншим об'єктам. Кожен об'єкт — своєрідний незалежний автомат з окремим призначенням та відповідальністю.

Спрощена схема алгоритму роботи програми представлена на рис 3.5.

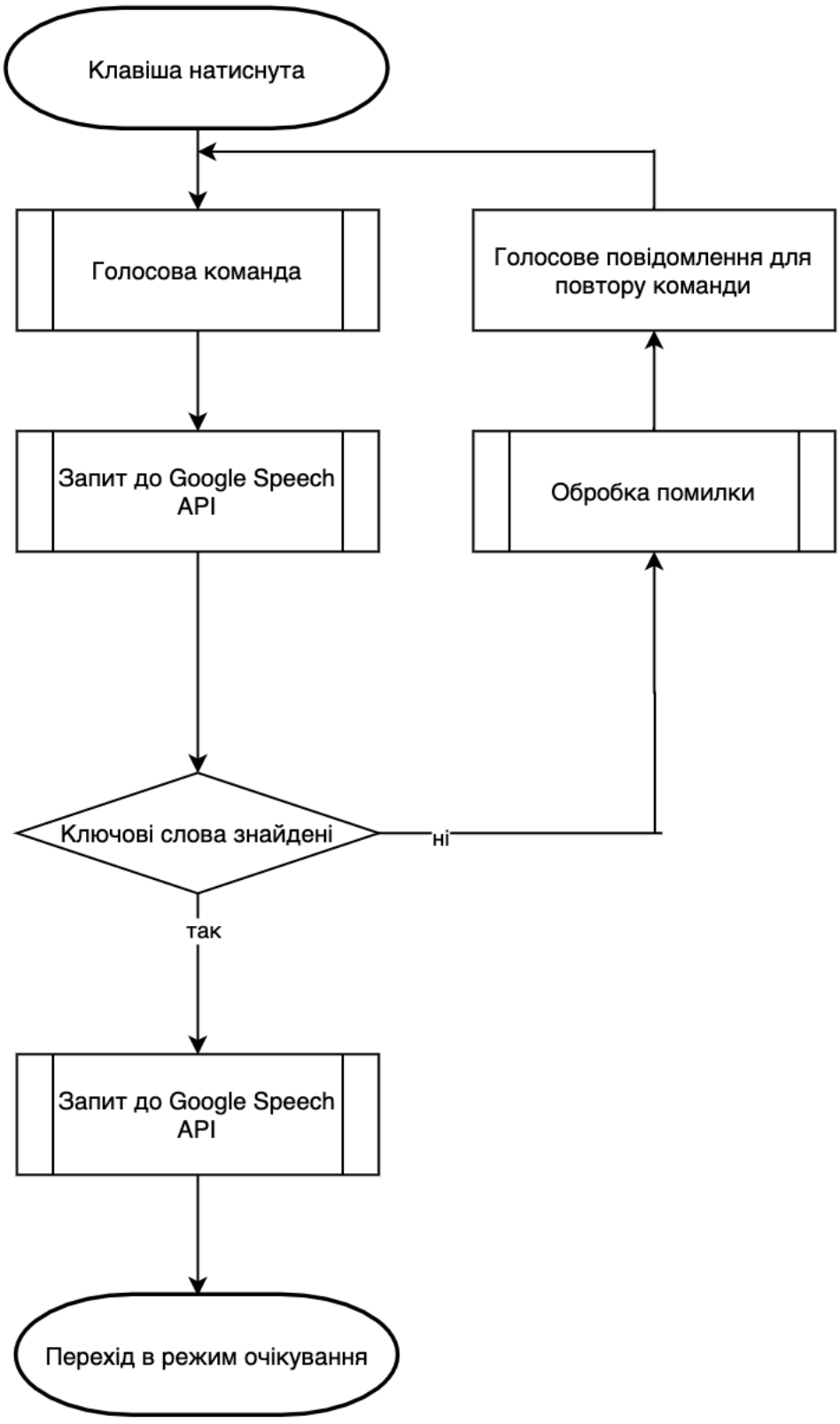


Рис. 3.5 – Спрощена схема алгоритму роботи програми

Більш детальна схема знаходиться у додатку 1.

Висновки до розділу 4

В розділі було розглянуто основні компоненти графічного фреймворку PyQT5, для їх наступного використання. Також було описано графічний інтерфейс програми та переваги об’єктно-орієнтованого програмування, складено відповідний алгоритм рішення задачі використовуючи цей підхід.

РОЗДІЛ 5
ПРОГРАМУВАННЯ ТА ТЕСТУВАННЯ

5.1 Розробка програми

Загалом розробка програми виконувалась в наступному порядку:

- розмітка каталогу з проектом, створення додаткових каталогів, створення файлової структури;
- створення глосарію;
- створення алгоритму додавання нових команд до глосарію та їх виконання
- створення та програмування двигуну;
- програмування механізму обробки помилок двигуна;
- створення та програмування механізму роботи з запитами;
- програмування механізму обробки помилок при роботі з Глосарієм;
- розробка головної форми.

Головна форма розроблялася в декілька етапів:

- програмування головної форми;
- написання власних функції для взаємодії с кнопками;
- розробка вигляду поточної форми;
- написання власних функції для взаємодії з глосарієм;
- програмування механізму регулярних виразів;
- перевірка працездатності та відладка коду.

Завершується розробка програми процесом тестуванням розробником, та формуванням відповідного інсталяційного файлу для полегшення процесу інсталяції програми.

UML-діаграма класів та структурна схема програми є у відповідних додатках В та Г.

5.2 Етапи налагодження

Перевірка та налагодження роботи програми відбувалася в кінці розробки кожної частини окремо.

Велика модульність програми дозволяє тестувати повністю завершену частину проекту одразу після її програмування.

Для виконання перевірки працездатності програми вона була випробувана в реальній ситуації.

Суть налагодження полягає у тому, що користувач розробляє систему тестів, за допомогою якої перевіряється робота програми у різних можливих режимах. Кожен тест має набір вхідних даних, для яких відомий результат. Тест намагаються вибрати так, щоб не тільки встановити сам факт помилки, але й локалізувати її, тобто виявити та звузити частину програми, що містить помилку.

5.3 Типи помилок

Для обробки помилок у функції розпізнавання мови та у функції роботи з глосарієм було створена (у кожному окремо) система обробки помилок.

Їх принцип роботи дуже схожий один з одним. Кожен раз після виклику власних функції перевіряється, чи була помилка, якщо була, то вона фіксується, додається власний опис з місцем виникнення помилки, та вона виводиться.

Якщо при виконанні функцій сталася помилка, то в вікні програми буде власний опис помилки, з вказанням найменування функції, яка викликала помилку, а також стандартний Python-опис помилки:

- UnknownValuteError опис – «Sorry, I couldn’t understand your speech. Can you repeat please?
- RequestError опис – «Google Speech Recognition Service is unavailable at the moment»;
- SyntaxError: invalid syntax опис – «Wrong operator»

Висновки до розділу 5

У цьому розділі було описано етапи розробки як самого алгоритму роботи програми, так і графічного інтерфейсу. Також було розглянуто варіанти можливих помилок програми, та процес налагодження для знаходження помилок.

ВИСНОВКИ

Метою дипломного проекту «Голосове управління комп'ютером на основі глосарію за допомогою алгоритмів розпізнання мови» є створення програми для використання у повсякденному житті.

Програмний продукт містить у собі:

- бібліотеки з алгоритмом розпізнавання голосу;
- алгоритм встановлення команд;
- алгоритм виконання команд;
- глосарій.

Склад програмного продукту:

- зручна візуальна оболонка;
- інструкція з експлуатації програми.

Об'єктом автоматизації є процес виконання вказаних голосових команд.

Під час роботи над даним продуктом мною було освоєно середовище розробки QtDesigner та мова програмування Python.

Програмний продукт може буду удосконалений шляхом додавання нових функцій та класів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Python[Електронний ресурс]. – Режим доступу: https://ru.wikipedia.org/wiki/Python

2. PyQt5 [Електронний ресурс]. – Режим доступу: https://ru.wikipedia.org/wiki/PyQt

3. QtDesigner [Електронний ресурс]. – Режим доступу: https://ru.wikipedia.org/wiki/Qt_Designer

4. Адитья Бхаргава «Грокаем алгоритмы»[Текст]

5. Swaroop C.H. «A Byte of Python»[Текст]

6. Сибаров, Ю.Г Охрана труда в вычислительных центрах[Текст]/ Ю.Г. Сибаров, Н.Н. Сколотнев Машиностроение 2009.

7. Розрахунок собівартості програмного продукту [Електронний ресурс] – Режим доступу: https://ua-referat.com/Розрахунок_собівартості_і_вартості_програмного_продукту_з_обліку_переривань_на_мові_Асемблер

8. R. Fielding, Architectural styles and the design of network-based software architectures, PhD Thesis. University of California, Irvine, Information and Computer Science Department. 2000.

9. H. McWilliam, Analysis tool web services, from the EMBL-EBI, Nucleic Acids Res., 2013, vol. 41 (pg. W597-W600).

10. Віртуальний помічник [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Віртуальний_помічник

11. What is Apple Siri [Електронний ресурс] – Режим доступу: https://www.pocket-lint.com/apps/news/apple/112346-what-is-siri-apple-s-personal-voice-assistant-explained

12. What is Microsoft Cortana [Електронний ресурс] – Режим доступу: https://searchenterprisedesktop.techtarget.com/definition/Cortana

13. What is Google Assistant [Електронний ресурс] – Режим доступу: https://www.pocket-lint.com/apps/news/google/137722-what-is-google-assistant-how-does-it-work-and-which-devices-offer-it

14. Python benefits [Електронний ресурс] – Режим доступу: https://www.invensis.net/blog/it/benefits-of-python-over-other-programming-languages/

15. Лутц М. Программирование на Python, том II, 4-е издание. – Пер. сангл. – СПб.: Символ-Плюс, 2011. – 992 с

16. Integrated development environment [Електронний ресурс] – Режим доступу: https://www.educba.com/what-is-ide/

17. QT story [Електронний ресурс] – Режим доступу: https://rtime.felk.cvut.cz/osp/prednasky/gui/the-qt-story/

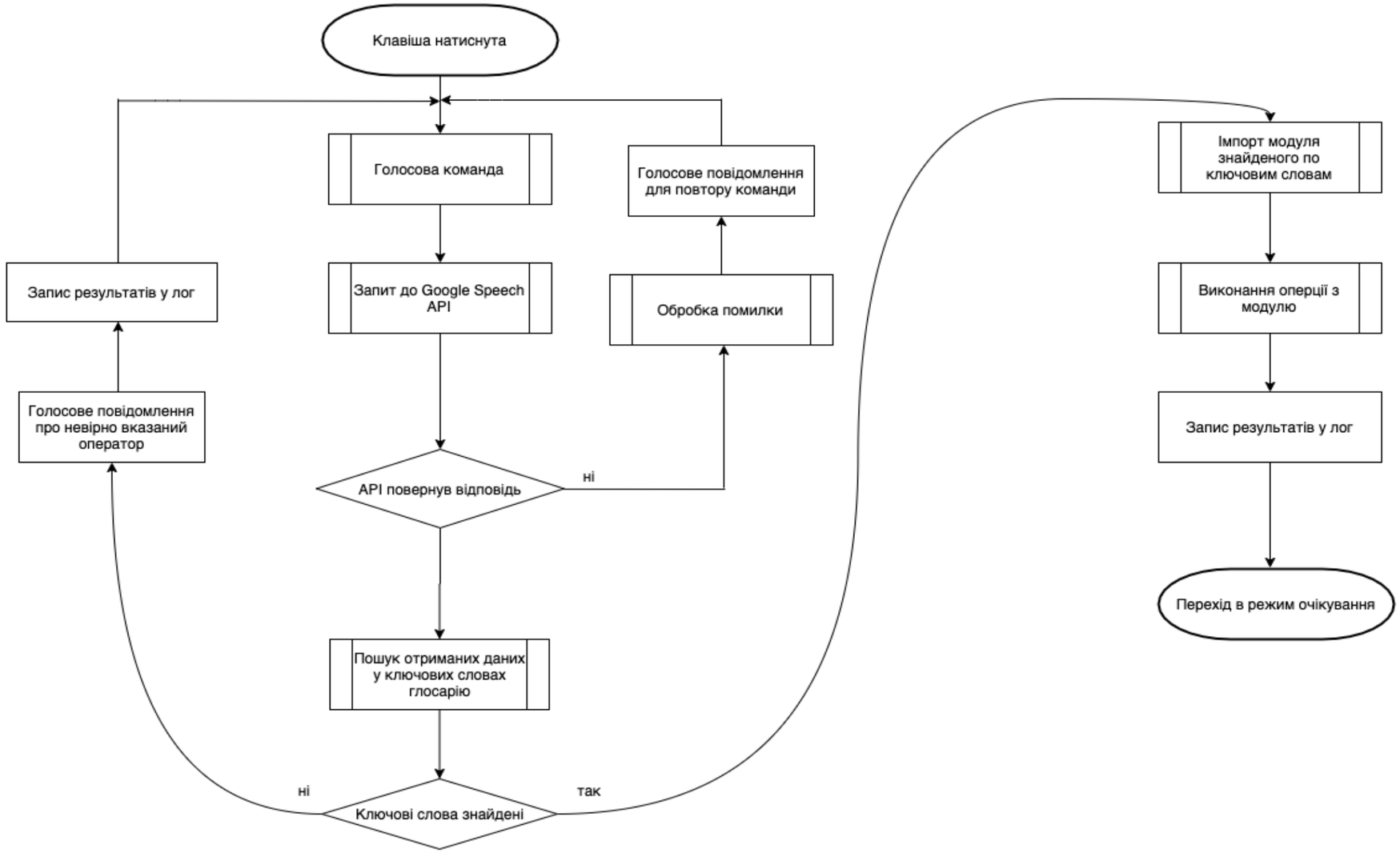
18. Linux benefits [Електронний ресурс] – Режим доступу: https://techdifferences.com/difference-between-linux-and-windows-operating-system.htl

19. PyInstaller bundler [Електронний ресурс] – Режим доступу: https://pypi.org/project/PyInstaller/

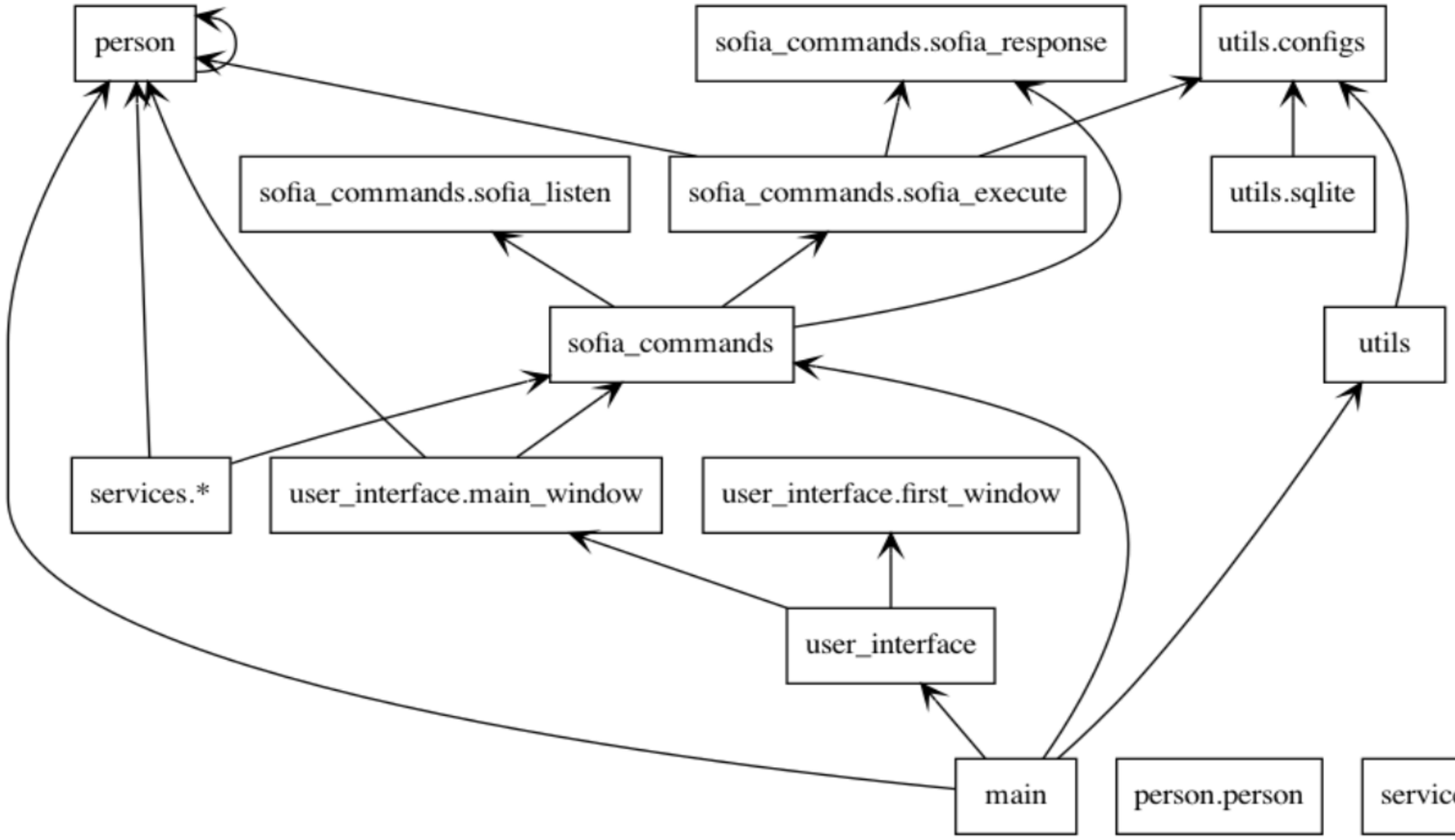
20. Object-oriented programming with python [Електронний ресурс] – Режим доступу: https://www.ntu.edu.sg/home/ehchua/programming/webprogramming/Python1a_OOP.html

21. SQLite vs MySQL vs PostgreSQL [Електронний ресурс] – Режим доступу: https://www.digitalocean.com/community/tutorials/sqlite-vs-mysql-vs-postgresql-a-comparison-of-relational-database-management-systems

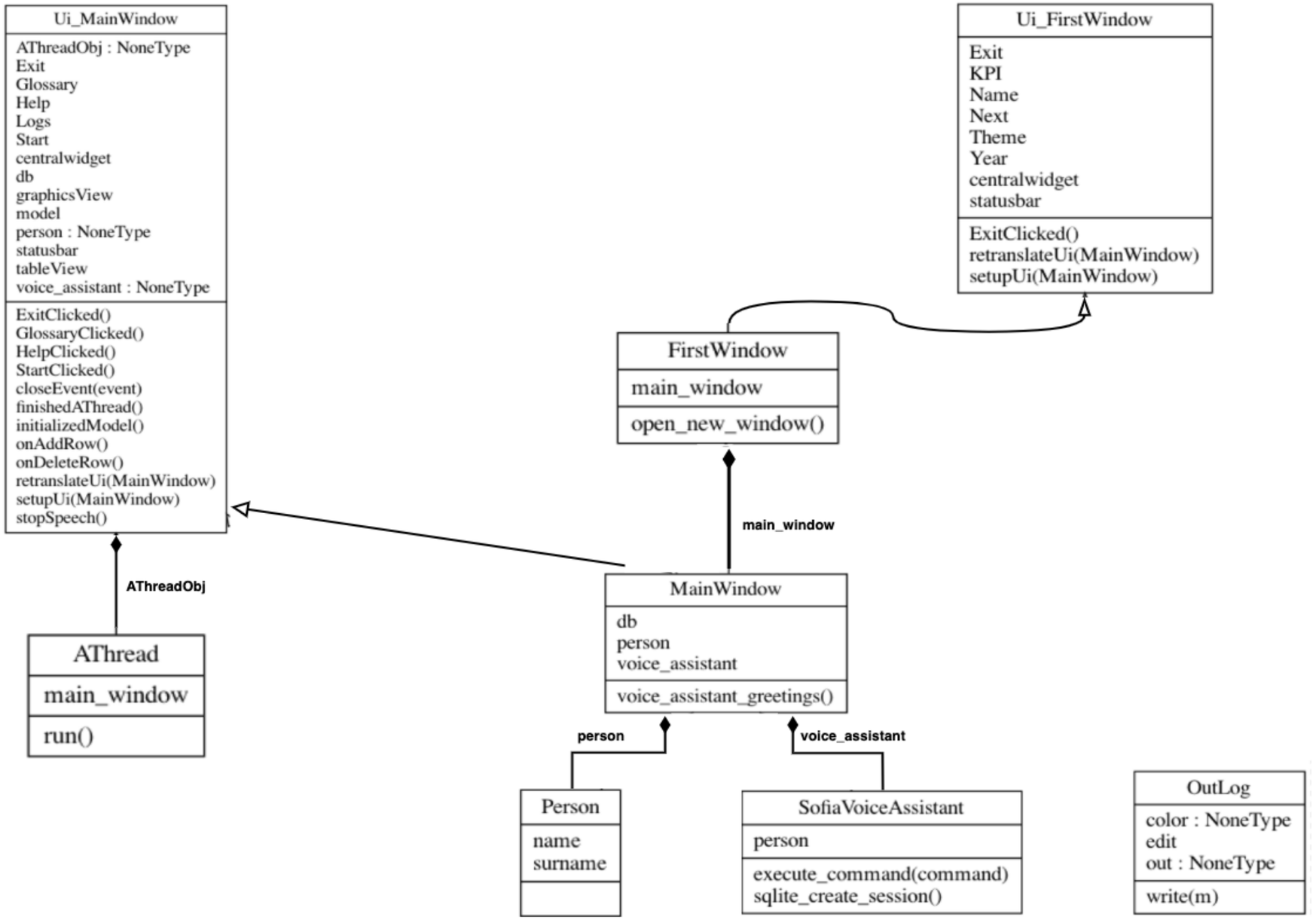
22. Comparint PostgreSQL vs MongoDB [Електронний ресурс] – Режим доступу: https://www.mongodb.com/compare/mongodb-postgresql



					ІАЛЦ.045430.004 Д1		
Зм.	Лист	№ докум.	Підп.	Дата	Принципова схема додатку голосового помічника	Літера	Маса / Масштаб
Розроб.		Дудченко І.В.					
Т.контр.						Лист 1	Листів
Н.контр.		Сімоненко В.П.					
Затв.							



					ІАЛЦ.045430.005 Д2			
Зм.	Лист	№ докум.	Підп.	Дата	Структурна схема додатку голосового помічника	Літера	Маса	Масштаб
Розроб.		Дудченко І.В.						
						Лист 2	Листів	
Т.контр.								
Н.контр.		Сімоненко В.П.						
Затв.								



					ІАЛЦ.045430.006 Д3		
						Літера	*Маса*	*Масштаб*
Зм.	*Лист*	*№ докум.*	*Підп.*	*Дата*	Функціональна схема алгоритму голосового помічника			
Розроб.		*Дудченко І.В.*						
						Лист 3	*Листів*	
Т.контр.								
Н.контр.		*Сімоненко В.П.*						
Затв.								

Текст програми

Графічний інтерфейс

main_window.py

```
import signal
import os

from PyQt5 import QtCore, QtGui, QtWidgets
from PyQt5.QtCore import Qt, QThread, pyqtSignal
from PyQt5.QtSql import QSqlTableModel, QSqlDatabase

from sofia_commands import listen_command
from sofia_commands import SofiaVoiceAssistant
from sofia_commands import engine_speak
from person import Person


class AThread(QThread):

    def __init__(self, main_window):
        self.main_window = main_window
        super().__init__()

    def run(self):
        engine_speak("eek", verbose=False)
        self.main_window.voice_assistant.execute_command(listen_command())


# noinspection PyPep8Naming,PyAttributeOutsideInit,PyTypeChecker
class Ui_MainWindow(object):
    person: Person = None
    voice_assistant: SofiaVoiceAssistant = None
    db: QSqlDatabase
    AThreadObj: QThread = None

    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(1127, 814)
        MainWindow.setStyleSheet("")
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")

        self.Exit = QtWidgets.QPushButton(self.centralwidget)
        self.Exit.setGeometry(QtCore.QRect(110, 640, 112, 32))
```

```
self.Exit.setObjectName("Exit")
self.Exit.clicked.connect(self.ExitClicked)
shortcutExit = QtWidgets.QShortcut(
    QtGui.QKeySequence("F"), self.Exit
)
shortcutExit.activated.connect(self.ExitClicked)
shortcutExit.setEnabled(True)

self.Help = QtWidgets.QPushButton(self.centralwidget)
self.Help.setGeometry(QtCore.QRect(110, 520, 112, 32))
self.Help.setObjectName("Help")
self.Help.clicked.connect(self.HelpClicked)

self.Glossary = QtWidgets.QPushButton(self.centralwidget)
self.Glossary.setGeometry(QtCore.QRect(970, 520, 112, 32))
self.Glossary.setObjectName("Glossary")
self.Glossary.clicked.connect(self.GlossaryClicked)

self.Start = QtWidgets.QPushButton(self.centralwidget)
self.Start.setGeometry(QtCore.QRect(970, 640, 112, 32))
self.Start.setObjectName("Start")
shortcutStart = QtWidgets.QShortcut(QtGui.QKeySequence("K"), self.Start)
shortcutStart.activated.connect(self.StartClicked)
shortcutStart.setEnabled(True)
self.Start.clicked.connect(self.StartClicked)

self.Logs = QtWidgets.QTextBrowser(self.centralwidget)
self.Logs.setGeometry(QtCore.QRect(230, 520, 731, 151))
self.Logs.setObjectName("Logs")

self.graphicsView = QtWidgets.QGraphicsView(self.centralwidget)
self.graphicsView.setGeometry(QtCore.QRect(230, 20, 731, 331))
self.graphicsView.setObjectName("graphicsView")


MainWindow.setCentralWidget(self.centralwidget)

self.statusbar = QtWidgets.QStatusBar(MainWindow)
self.statusbar.setObjectName("statusbar")

MainWindow.setStatusBar(self.statusbar)

self.retranslateUi(MainWindow)
QtCore.QMetaObject.connectSlotsByName(MainWindow)
```

```
def retranslateUi(self, MainWindow):
    _translate = QtCore.QCoreApplication.translate
    MainWindow.setWindowTitle(_translate("MainWindow", "Головне вікно"))
    self.Exit.setText(_translate("MainWindow", "Вихід"))
    self.Help.setText(_translate("MainWindow", "Довідка"))
    self.Glossary.setText(_translate("MainWindow", "Глосарій"))
    self.Start.setText(_translate("MainWindow", "Запуск"))

def HelpClicked(self):
    my_dialog = QtWidgets.QDialog(self)
    my_dialog.setGeometry(500, 500, 500, 500)
    qr = my_dialog.frameGeometry()
    cp = QtWidgets.QDesktopWidget().availableGeometry().center()
    qr.moveCenter(cp)
    my_dialog.move(qr.topLeft())
    my_dialog.setWindowTitle("Довідка")
    l1 = QtWidgets.QLabel()
    l1.setText('"1. Скажіть help me для отримання можливих команд з
    глосарію\n 2. Для активації голосового помічника натисніть кнопку
    "Запуск" або скористуйтесь клавішею K. \n 3. Для перегляду та
    редагування глосарію натисніть кнопку "Глосарій"\n 4. Розпізнаний
    текст виводиться у ділоговому вікні в головному меню додатку"')
    l1.setAlignment(Qt.AlignCenter)
    vbox = QtWidgets.QVBoxLayout()
    vbox.addWidget(l1)
    l1.setOpenExternalLinks(True)
    my_dialog.setLayout(vbox)
    my_dialog.exec_()

def ExitClicked(self):
    reply = QtWidgets.QMessageBox.question(self, 'Message',
                                           "Ви впевненні що хочете вийти?",
                                           QtWidgets.QMessageBox.Yes |
                                           QtWidgets.QMessageBox.No,
                                           QtWidgets.QMessageBox.No)


    if reply == QtWidgets.QMessageBox.Yes:
        QtCore.QCoreApplication.instance().quit()

def StartClicked(self):
    if self.AThreadObj is None:
        self.AThreadObj = AThread(self)
        self.AThreadObj.finished.connect(self.finishedAThread)
        self.AThreadObj.start()
```

```
            self.Start.setText("Стоп")
        else:
            # self.AThreadObj.terminate()
            self.AThreadObj = None
            self.stopSpeech()
            self.Start.setText("Запуск")

    def finishedAThread(self):
        self.AThreadObj = None
        self.Start.setText("Запуск")

    def initializedModel(self):
        self.model.setTable("glossary_modules")
        self.model.setEditStrategy(QSqlTableModel.OnFieldChange)
        self.model.select()
        self.model.setHeaderData(0, Qt.Horizontal, "Keywords")
        self.model.setHeaderData(1, Qt.Horizontal, "Module name")

    def onAddRow(self):
        self.model.insertRows(self.model.rowCount(), 1)
        self.model.submit()

    def onDeleteRow(self):
        self.model.removeRow(self.tableView.currentIndex().row())
        self.model.submit()
        self.model.select()

    def closeEvent(self, event):
        self.db.close()

    def GlossaryClicked(self):
        self.model = QSqlTableModel()
        self.initializedModel()

        my_dialog = QtWidgets.QDialog(self)

        self.tableView = QtWidgets.QTableView()
        self.tableView.setModel(self.model)


        layout = QtWidgets.QVBoxLayout()

        addButton = QtWidgets.QPushButton("add")
        deleteButton = QtWidgets.QPushButton("delete")
```

```
        hLayout = QtWidgets.QHBoxLayout()
        hLayout.addWidget(addButton)
        hLayout.addWidget(deleteButton)

        layout.addWidget(self.tableView)
        layout.addLayout(hLayout)

        addButton.clicked.connect(self.onAddRow)
        deleteButton.clicked.connect(self.onDeleteRow)

        my_dialog.setLayout(layout)
        my_dialog.resize(600, 400)
        my_dialog.exec_()

    def stopSpeech(self):
        for line in os.popen("ps ax | grep say | grep -v grep"):
            fields = line.split()
            pid = fields[0]
            try:
                os.kill(int(pid), signal.SIGKILL)
            except ProcessLookupError:
                pass
```

first_window.py

```
from PyQt5 import QtCore, QtWidgets


# noinspection PyTypeChecker,PyPep8Naming,PyAttributeOutsideInit
class Ui_FirstWindow(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(800, 600)
        MainWindow.setStyleSheet("")
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")

        self.KPI = QtWidgets.QTextBrowser(self.centralwidget)
        self.KPI.setEnabled(True)
        self.KPI.setGeometry(QtCore.QRect(250, 30, 311, 61))
        self.KPI.setFocusPolicy(QtCore.Qt.StrongFocus)
        self.KPI.setAutoFillBackground(False)
        self.KPI.setStyleSheet("")
```

```
self.KPI.setReadOnly(True)
self.KPI.setObjectName("KPI")

self.Theme = QtWidgets.QTextBrowser(self.centralwidget)
self.Theme.setEnabled(True)
self.Theme.setGeometry(QtCore.QRect(250, 90, 311, 81))
self.Theme.setAcceptDrops(True)
self.Theme.setObjectName("Theme")

self.Name = QtWidgets.QTextBrowser(self.centralwidget)
self.Name.setGeometry(QtCore.QRect(560, 230, 241, 131))
self.Name.setObjectName("Name")

self.Year = QtWidgets.QTextBrowser(self.centralwidget)
self.Year.setGeometry(QtCore.QRect(350, 510, 111, 31))
self.Year.setObjectName("Year")

self.Exit = QtWidgets.QPushButton(self.centralwidget)
self.Exit.setGeometry(QtCore.QRect(10, 540, 112, 32))
self.Exit.setObjectName("Exit")
self.Exit.clicked.connect(self.ExitClicked)

self.Next = QtWidgets.QPushButton(self.centralwidget)
self.Next.setGeometry(QtCore.QRect(680, 540, 112, 32))
self.Next.setObjectName("Next")

self.Theme.raise_()
self.Name.raise_()
self.Year.raise_()
self.Exit.raise_()
self.Next.raise_()
self.KPI.raise_()

MainWindow.setCentralWidget(self.centralwidget)

self.statusbar = QtWidgets.QStatusBar(MainWindow)
self.statusbar.setObjectName("statusbar")

MainWindow.setStatusBar(self.statusbar)

self.retranslateUi(MainWindow)
QtCore.QMetaObject.connectSlotsByName(MainWindow)
```

```
    def retranslateUi(self, MainWindow):
        _translate = QtCore.QCoreApplication.translate
        MainWindow.setWindowTitle(
            _translate("MainWindow", "Початкове вікно"))

        self.Exit.setText(_translate("MainWindow", "Вихід"))
        self.Next.setText(_translate("MainWindow", "Далі"))

    def ExitClicked(self):
        reply = QtWidgets.QMessageBox.question(self, 'Message',
                                               "Ви впевненні що хочете вийти?",
                                               QtWidgets.QMessageBox.Yes |
                                               QtWidgets.QMessageBox.No,
                                               QtWidgets.QMessageBox.No)
        if reply == QtWidgets.QMessageBox.Yes:
            QtCore.QCoreApplication.instance().quit()
```

Модуль ядра голосового помічника

sofia_execute.py

```
import importlib
import sqlite3

from dataclasses import dataclass

from person import Person
from utils.configs import PATH_TO_SQLITE
from sofia_commands.sofia_response import engine_speak


@dataclass
class SofiaVoiceAssistant:
    person: Person

    @staticmethod
    def sqlite_create_session():
        conn = sqlite3.connect(PATH_TO_SQLITE)
        cursor = conn.cursor()
        return cursor

    def execute_command(self, command):
        db_session = self.sqlite_create_session()
```

```
        operator_is_valid = False

        for row in db_session.execute(f"SELECT * FROM glossary_modules"):
            list_of_keywords = row[0].split(',')


            if any(keyword in command for keyword in list_of_keywords):
                # TODO: check other results
                module_name = row[1]
                full_module_name = 'services.' + module_name
                service_module = importlib.import_module(full_module_name)
                service_module.execute(command, self.person)
                operator_is_valid = True

        if not operator_is_valid:
            engine_speak("Wrong Operator")
```

sofia_response.py

```
import os
from typing import Any


def engine_speak(text: Any, verbose: bool = True):
    """
    Sofia says text
    :param text: text to be said
    :param verbose print text to stdout or not
    :return: None
    """
    if verbose:
        print(text + '\n')

    if not isinstance(text, str):
        text = str(text)

    for _ in text.splitlines():
        os.system("say " + text)
```

sofia_listen.py

```
import speech_recognition as sr

def listen_command():
    """
    Listen to audio inout and fetch commands from google cloud speech API
    :return: command: str - text command
    """
    r = sr.Recognizer()

    with sr.Microphone() as source:
        print('Say something...')
        r.pause_threshold = 1
        r.adjust_for_ambient_noise(source, duration=1)
        audio = r.listen(source)

    try:
        command = r.recognize_google(audio).lower()
        print('You said: ' + command + '\n')

    # loop back to continue to listen for
    # commands if unrecognizable speech is received
    except sr.UnknownValueError:
        print('Sorry, I couldnt understand your speech. Can you repeat please?')
        command = listen_command()

    return command
```

Приклади модулів виконання для глоссарію

open_website.py

```
import re
import webbrowser

from sofia_commands import engine_speak
from person import Person

def execute(voice_input: str, person: Person):
    reg_ex = re.search('([aA-zZ])+(.com|.ua|.ru)', voice_input)
    if reg_ex:
        domain = reg_ex.group()
        print(domain)
```

```
        url = 'https://www.' + domain
        webbrowser.open(url)
        engine_speak(
            f'The website you have requested '
            f'has been opened for you {person.name}.'
        )
    else:
        pass
```

change_wallpaper.py

```
import os
import json
import subprocess
from appscript import app, mactypes

from urllib.request import urlopen, urlretrieve

from sofia_commands import engine_speak
from person import Person


def execute(voice_input: str, person: Person):
    folder = os.getcwd()

    for the_file in os.listdir(folder):
        file_path = os.path.join(folder, the_file)
        try:
            if os.path.isfile(file_path):
                os.unlink(file_path)
        except Exception as e:
            print(e)

    api_key = ************************************

    # pic from unspalsh.com
    url = 'https://api.unsplash.com/photos/random?client_id=' + api_key
    f = urlopen(url)
    json_string = f.read()
    f.close()
    parsed_json = json.loads(json_string)
    photo = parsed_json['urls']['full']
```

```
    # Location where we download the image to.
    urlretrieve(photo, ‘a.png’)

    app('Finder').desktop_picture.set(mactypes.File(‘a.png’))

    subprocess.call(["killall Dock"], shell=True)

    engine_speak('wallpaper changed successfully')
```

play_song.py

```
import os
import platform
import sys
from urllib.request import urlopen

import youtube_dl
import vlc
from bs4 import BeautifulSoup as soup
from PyQt5 import QtWidgets, QtGui, QtCore

from sofia_commands import engine_speak, listen_command
from person import Person

class Player(QtWidgets.QMainWindow):
    """A simple Media Player using VLC and Qt
    """

    def __init__(self, master=None):
        QtWidgets.QMainWindow.__init__(self, master)
        self.setWindowTitle("Media Player")

        # Create a basic vlc instance
        self.instance = vlc.Instance()

        self.media = None

        # Create an empty vlc media player
        self.mediaplayer = self.instance.media_player_new()

        self.create_ui()
        self.is_paused = False
```

```
def create_ui(self):
    """Set up the user interface, signals & slots
    """
    self.widget = QtWidgets.QWidget(self)
    self.setCentralWidget(self.widget)

    # In this widget, the video will be drawn
    if platform.system() == "Darwin": # for MacOS
        self.videoframe = QtWidgets.QMacCocoaViewContainer(0)
    else:
        self.videoframe = QtWidgets.QFrame()

    self.palette = self.videoframe.palette()
    self.palette.setColor(QtGui.QPalette.Window, QtGui.QColor(0, 0, 0))
    self.videoframe.setPalette(self.palette)
    self.videoframe.setAutoFillBackground(True)

    self.positionslider = QtWidgets.QSlider(QtCore.Qt.Horizontal, self)
    self.positionslider.setToolTip("Position")
    self.positionslider.setMaximum(1000)
    self.positionslider.sliderMoved.connect(self.set_position)
    self.positionslider.sliderPressed.connect(self.set_position)

    self.hbuttonbox = QtWidgets.QHBoxLayout()
    self.playbutton = QtWidgets.QPushButton("Play")
    self.hbuttonbox.addWidget(self.playbutton)
    self.playbutton.clicked.connect(self.play_pause)

    self.stopbutton = QtWidgets.QPushButton("Stop")
    self.hbuttonbox.addWidget(self.stopbutton)
    self.stopbutton.clicked.connect(self.stop)

    self.hbuttonbox.addStretch(1)
    self.volumeslider = QtWidgets.QSlider(QtCore.Qt.Horizontal, self)
    self.volumeslider.setMaximum(100)
    self.volumeslider.setValue(self.mediaplayer.audio_get_volume())
    self.volumeslider.setToolTip("Volume")
    self.hbuttonbox.addWidget(self.volumeslider)
    self.volumeslider.valueChanged.connect(self.set_volume)

    self.vboxlayout = QtWidgets.QVBoxLayout()
    self.vboxlayout.addWidget(self.videoframe)
    self.vboxlayout.addWidget(self.positionslider)
    self.vboxlayout.addLayout(self.hbuttonbox)
```

```
        self.widget.setLayout(self.vboxlayout)

        menu_bar = self.menuBar()

        # File menu
        file_menu = menu_bar.addMenu("File")

        # Add actions to file menu
        open_action = QtWidgets.QAction("Load Video", self)
        close_action = QtWidgets.QAction("Close App", self)
        file_menu.addAction(open_action)
        file_menu.addAction(close_action)

        open_action.triggered.connect(self.open_file)
        close_action.triggered.connect(sys.exit)

        self.timer = QtCore.QTimer(self)
        self.timer.setInterval(100)
        self.timer.timeout.connect(self.update_ui)

    def play_pause(self):
        """Toggle play/pause status
        """
        if self.mediaplayer.is_playing():
            self.mediaplayer.pause()
            self.playbutton.setText("Play")
            self.is_paused = True
            self.timer.stop()
        else:
            if self.mediaplayer.play() == -1:
                self.open_file()
                return

            self.mediaplayer.play()
            self.playbutton.setText("Pause")
            self.timer.start()
            self.is_paused = False

    def stop(self):
        """Stop player
        """
        self.mediaplayer.stop()
        self.playbutton.setText("Play")
```

```
    def open_file(self):
        """Open a media file in a MediaPlayer
        """

        dialog_txt = "Choose Media File"
        filename = QtWidgets.QFileDialog.getOpenFileName(self, dialog_txt,
os.path.expanduser('~'))
        if not filename:
            return

        # getOpenFileName returns a tuple, so use only the actual file name
        self.media = self.instance.media_new(filename[0])

        # Put the media in the media player
        self.mediaplayer.set_media(self.media)

        # Parse the metadata of the file
        self.media.parse()

        # Set the title of the track as window title
        self.setWindowTitle(self.media.get_meta(0))

        # The media player has to be 'connected' to the QFrame (otherwise the
        # video would be displayed in it's own window). This is platform
        # specific, so we must give the ID of the QFrame (or similar object) to
        # vlc. Different platforms have different functions for this
        if platform.system() == "Linux": # for Linux using the X Server
            self.mediaplayer.set_xwindow(int(self.videoframe.winId()))
        elif platform.system() == "Windows": # for Windows
            self.mediaplayer.set_hwnd(int(self.videoframe.winId()))
        elif platform.system() == "Darwin": # for MacOS
            self.mediaplayer.set_nsobject(int(self.videoframe.winId()))

        self.play_pause()

    def set_volume(self, volume):
        """Set the volume
        """
        self.mediaplayer.audio_set_volume(volume)

    def set_position(self):
        """Set the movie position according to the position slider.
        """
```

```
        # The vlc MediaPlayer needs a float value between 0 and 1, Qt uses
        # integer variables, so you need a factor; the higher the factor, the
        # more precise are the results (1000 should suffice).

        # Set the media position to where the slider was dragged
        self.timer.stop()
        pos = self.positionslider.value()
        self.mediaplayer.set_position(pos / 1000.0)
        self.timer.start()

    def update_ui(self):
        """Updates the user interface"""

        # Set the slider's position to its corresponding media position
        # Note that the setValue function only takes values of type int,
        # so we must first convert the corresponding media position.
        media_pos = int(self.mediaplayer.get_position() * 1000)
        self.positionslider.setValue(media_pos)

        # No need to call this function if nothing is played
        if not self.mediaplayer.is_playing():
            self.timer.stop()

            # After the video finished, the play button stills shows "Pause",
            # which is not the desired behavior of a media player.
            # This fixes that "bug".
            if not self.is_paused:
                self.stop()

def execute(voice_input: str, person: Person):
    path = '/Users/idudchenko/Documents/videos/'
    folder = path
    for the_file in os.listdir(folder):
        file_path = os.path.join(folder, the_file)
        try:
            if os.path.isfile(file_path):
                os.unlink(file_path)
        except Exception as e:
            print(e)

    engine_speak(f'What song shall I play ?')
    mysong = listen_command()
    if mysong:
        flag = 0
```

```
    url = "https://www.youtube.com/results?search_query=" + mysong.replace(
        ' ', '+')
    response = urlopen(url)
    html = response.read()
    soup1 = soup(html, "lxml")
    url_list = []
    for vid in soup1.findAll(attrs={'class': 'yt-uix-tile-link'}):
        if ('https://www.youtube.com' + vid['href']).startswith(
                "https://www.youtube.com/watch?v="):
            flag = 1
            final_url = 'https://www.youtube.com' + vid['href']
            url_list.append(final_url)

    try:
        url = url_list[0]
    except IndexError:
        pass

    ydl_opts = {'outtmpl': 'song1.mp4'}

    os.chdir(path)
    with youtube_dl.YoutubeDL(ydl_opts) as ydl:
        ydl.download([url])

    if flag == 0:
        engine_speak('I have not found anything in Youtube ')

    app = QtWidgets.QApplication(sys.argv)
    player = Player()
    player.show()
    player.resize(1280, 720)
    player_ = vlc.MediaPlayer(path+'song1.mp4')
    player_.set_nsobject((int(player.videoframe.winId())))
    player_.play()
    sys.exit(app.exec_())
```

Головна програма

main.py

```
import sys

from PyQt5 import QtWidgets
from PyQt5.QtCore import QTimer
```

```
from PyQt5 import QtGui
from PyQt5.QtSql import QSqlDatabase

from person import Person
from sofia_commands import SofiaVoiceAssistant
from sofia_commands import engine_speak

from user_interface import Ui_FirstWindow
from user_interface import Ui_MainWindow
from utils import PATH_TO_SQLITE

# noinspection PyUnboundLocalVariable
class OutLog:
    def __init__(self, edit, out=None, color=None):
        self.edit = edit
        self.out = None
        self.color = color

    def write(self, m):
        if self.color:
            tc = self.edit.textColor()
            self.edit.setTextColor(self.color)

        self.edit.moveCursor(QtGui.QTextCursor.End)
        self.edit.insertPlainText(m)

        if self.color:
            self.edit.setTextColor(tc)

        if self.out:
            self.out.write(m)

# noinspection PyAttributeOutsideInit
class FirstWindow(QtWidgets.QMainWindow, Ui_FirstWindow):
    def __init__(self):
        super().__init__()
        self.setupUi(self)

        self.Next.clicked.connect(self.open_new_window)

    def open_new_window(self):
        self.main_window = MainWindow()
        self.main_window.show()
        self.close()
```

```
# noinspection PyAttributeOutsideInit
class MainWindow(QtWidgets.QMainWindow, Ui_MainWindow):
    def __init__(self):
        super().__init__()
        self.person = Person(name='Igor', surname='Dudchenko')
        self.voice_assistant = SofiaVoiceAssistant(self.person)
        self.db = QSqlDatabase.addDatabase("QSQLITE")
        self.db.setDatabaseName(PATH_TO_SQLITE)
        self.db.open()

        self.setupUi(self)

        # waits for this to finish until gui displayed
        QTimer.singleShot(1, self.voice_assistant_greetings)

        sys.stdout = OutLog(self.Logs, sys.stdout)
        sys.stderr = OutLog(self.Logs, sys.stderr, QtGui.QColor(255, 0, 0))

    def voice_assistant_greetings(self):
        engine_speak(f'Hi {self.person.name} '
                     f'{self.person.surname}, '
                     f'I am Sofia and I am your personal voice assistant. '
                     f'Please give a command or say "help me " '
                     f'and I will tell you what all I can do for you.')


def main():
    app = QtWidgets.QApplication(sys.argv)
    window = FirstWindow()
    window.show()
    app.exec_()


if __name__ == '__main__':
    main()
```